

MacTech®

The Journal of Macintosh Technology and Development

HIObject: The Carbon Object Model by Ed Voas

*Learn about the new Toolbox object
model introduced in Mac OS X 10.2*



\$8.95 US
\$12.95 Canada
ISSN 1067-8360
Printed in U.S.A.



We got
top marks
for our
software.



You can too.

Revolution™
The Solution for Software Development
In enterprise, business and education

REVOLUTION™

Limitless possibilities

With Revolution, the solutions you can create are endless.

True cross-platform development

With the click of a button, you can build applications for Macintosh (Classic and OS X), Windows, Linux, and Unix.

Increased productivity

The powerful interface builder and easy-to-use programming language speed up all the essentials of software development, leaving you with more time to focus on your project.

Databases, multimedia, Internet applications, external libraries and more

Revolution supports everything you need: SQL databases, streaming media, control of QuickTime, QTVR and graphics, CGI processing, Internet protocols, and more.

FREE Trial Version

www.runrev.com



MacUser UK - 5 mice

©2002 Runtime Revolution Limited. All rights reserved. Runtime Revolution, the Runtime Revolution logo and Revolution are trademarks of Runtime Revolution Limited, registered in the United Kingdom. All other trademarks are the property of their respective owners.



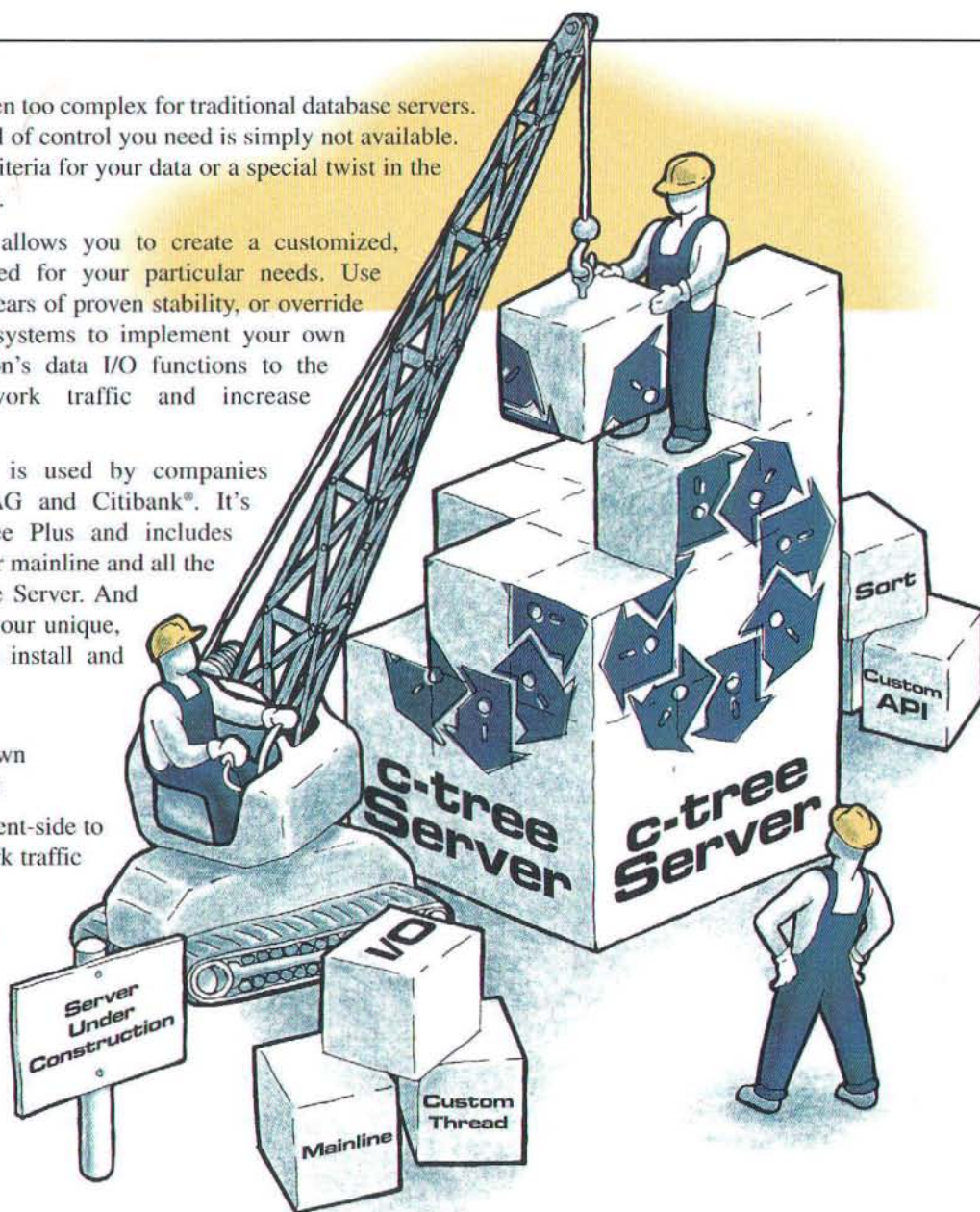
CUSTOMIZE YOUR DATABASE SERVER WITH THE C-TREE[®] SERVER SDK

Today's database demands are often too complex for traditional database servers. The functionality and precise level of control you need is simply not available. Perhaps you need alternate sort criteria for your data or a special twist in the threading or communication logic.

FairCom's c-tree[®] Server SDK allows you to create a customized, industrial-strength server designed for your particular needs. Use FairCom's kernel, with over 20 years of proven stability, or override functionality within specific subsystems to implement your own subtleties. Move your application's data I/O functions to the server-side to decrease network traffic and increase performance!

FairCom's c-tree Server SDK is used by companies worldwide such as Software AG and Citibank[®]. It's integrated seamlessly into c-tree Plus and includes complete source code to the server mainline and all the interface subsystems to the c-tree Server. And best of all, once you've created your unique, customized server, it is easy to install and administer: no DBA required!

- Enhance our server with your own custom server-side functionality
- Move functionality from the client-side to the server-side to reduce network traffic and increase performance
- Modify or replace entire server subsystems
- Complete source for the server mainline, key server subsystems, and client-side
- Flexible OEM licensing



Visit www.faircom.com/ep/mt/sdk today to take control of your server!



FairCom[®]
www.faircom.com

USA	573.445.6833
EUROPE	+39.035.773.464
JAPAN	+81.59.229.7504
BRAZIL	+55.11.3872.9802

DBMS Since 1979 • 800.234.8180 • info@faircom.com

Other company and product names are registered trademarks or trademarks of their respective owners.

© 2002 FairCom Corporation

XSERVE SCREAMS.

(FOR AN ENTERPRISE CLASS DATABASE.)

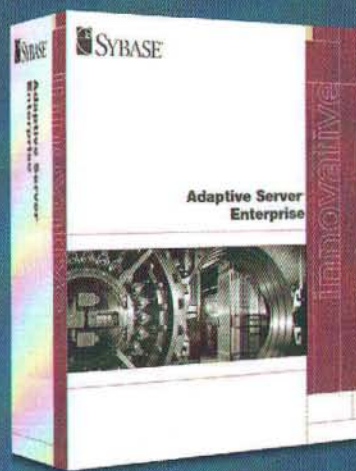


An insanely great enterprise server deserves an insanely great enterprise database. Sybase Adaptive Server™ Enterprise 12.5. Wall Street's preferred platform for mission-critical, transaction-intensive

SYBASE e-BUSINESS SOFTWARE. EVERYTHING

THE STRAIGHT GOODS ON DATABASES.

WE ANSWER THE CALL.



enterprise applications. Now available to you on Apple Xserve™ with Mac OS X Server software. Ready to scream for joy? Check it out at www.sybase.com/mac



WORKS BETTER WHEN EVERYTHING WORKS TOGETHER.™

©2002 Sybase, Inc. All rights reserved. All trademarks are the property of their respective owners.

How To Communicate With Us

In this electronic age, the art of communication has become both easier and more complicated. Is it any surprise that we prefer **e-mail**?

If you have any questions, feel free to call us at 805/494-9797 or fax us at 805/494-9798.

If you would like a subscription or need customer service, feel free to contact Developer Depot Customer Service at 877-MACTECH

DEPARTMENTS

Orders, Circulation, & Customer Service

Press Releases

Ad Sales

Editorial

Programmer's Challenge

Online Support

Accounting

Marketing

General

Web Site (articles, info, URLs and more...)

E-Mail/URL

cust_service@mactech.com

press_releases@mactech.com

ad_sales@mactech.com

editorial@mactech.com

prog_challenge@mactech.com

online@mactech.com

accounting@mactech.com

marketing@mactech.com

info@mactech.com

http://www.mactech.com

The MacTech Editorial Staff

Publisher • Neil Ticktin

Managing Editor • Jessica Stubblefield

Online Editor • Jeff Clites

Regular Columnists

Getting Started – Networking

by John C. Welch

Programmer's Challenge

by Bob Boonstra

MacTech Online

by Jeff Clites

Regular Contributors

Vicki Brown, Andrew Stone,
Tim Monroe, Erick Tejkowski,
Kas Thomas, Will Porter,
Paul E. Sevinç, Tom Djajadiningrat,
and Jordan Dea-Mattson

MacTech's Board of Advisors

Dave Mark, Jordan Dea-Mattson,
Jim Straus, Jon Wiederspan,
and Eric Gundrum

MacTech's Contributing Editors

- Michael Brian Bentley
- Derek J. Burks
- Tantek Çelik, Microsoft Corporation
- Marshall Clow
- John. C. Daub
- Tom Djajadiningrat
- Bill Doerrfeld, Blueworld
- Andrew S. Downs
- Richard V. Ford, Packeteer
- Gordon Garb, Cobalt Networks
- Lorca Hanns, San Francisco State University
- Ilene Hoffman
- Chris Kilbourn, Digital Forest
- Scott Knaster, Microsoft
- Peter N. Lewis, Stairways Software
- Rich Morin
- John O'Fallon, Maxum Development
- Avi Rappoport, Search Tools Consulting
- Ty Shipman, Kagi
- Chuck Shotton, BIAP Systems
- Cal Simone, Main Event Software
- Steve Sisak, Codewell Corporation
- Andrew C. Stone, www.stone.com
- Michael Swan, Neon Software
- Chuck Von Rospach, Plaidworks
- Bill von Hagen
- Eric Zelenka

Xplain Corporation Staff

Chief Executive Officer • Neil Ticktin

President • Andrea J. Sniderman

Controller • Michael Friedman

Production Manager • Jessica Stubblefield

Production/Layout • Darryl Smart

Marketing Manager • Nick DeMello

Events Manager • Susan M. Worley

Network Administrator • David Breffitt

Accounting • Jan Webber, Marcie Moriarty

Customer Relations • Laura Lane

Susan Pomrantz

Shipping/Receiving • Joel Licardie

Board of Advisors • Steven Geller, Blake Park,
and Alan Carsrud

All contents are Copyright 1984-2002 by Xplain Corporation. All rights reserved. MacTech and Developer Depot are registered trademarks of Xplain Corporation. RadGad, Useful Gifts and Gadgets, Xplain, DevDepot, Depot, The Depot, Depot Store, Video Depot, Movie Depot, Palm Depot, Game Depot, Flashlight Depot, Explain It, MacDev-1, THINK Reference, NetProfessional, NetProLive, JavaTech, WebTech, BeTech, LinuxTech, MacTech Central and the MacTutorMan are trademarks or service marks of Xplain Corporation. Sprocket is a registered trademark of eSprocket Corporation. Other trademarks and copyrights appearing in this printing or software remain the property of their respective holders.

MacTech Magazine (ISSN: 1067-8360 / USPS: 010-227) is published monthly by Xplain Corporation, 850-P Hampshire Road, Westlake Village, CA 91361-2800. Voice: 805/494-9797, FAX: 805/494-9798. Domestic subscription rates are \$47.00 per year. Canadian subscriptions are \$59.00 per year. All other international subscriptions are \$97.00 per year. Domestic source code disk subscriptions are \$77 per year. All international disk subscriptions are \$97.00 a year. Please remit in U.S. funds only. Periodical postage is paid at Thousand Oaks, CA and at additional mailing office.

POSTMASTER: Send address changes to **MacTech Magazine**, P.O. Box 5200, Westlake Village, CA 91359-5200.

C o n t e n t s

October 2002 • Volume 18, Issue 10

VIEWPOINT

- 6 **User Solutions to Pesky
Trash Problems**
by Ilene Hoffman

COVER STORY

- 12 **HIObject:
The Carbon Object Model**
Learn about the new Toolbox
object model introduced in Mac
OS X 10.2
By Dan Wood, Alameda CA



MAC OS X

- 8 **When Wild Animals Bite
Or How To Workaround Cocoa bugs**
by Andrew C. Stone

COCOA DEVELOPEMENT

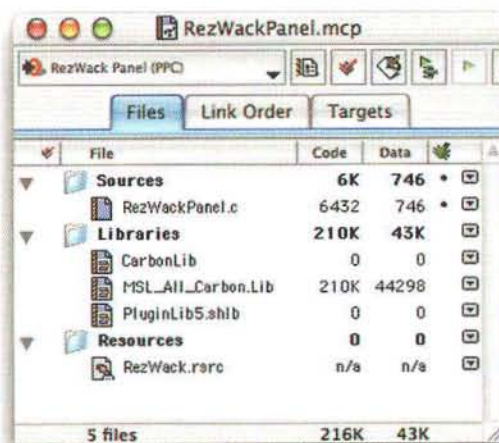
- 62 **Table Techniques Taught
Tastefully (part 2)**
Using NSTableView for Real-World Applications
By Dan Wood, Alameda CA

REVIEW

- 78 **Star Wars:
Galatic Battleground**

QUICKTIME TOOLKIT

- 24 **Human Resources**
*Adding Macintosh Resources to a Windows
QuickTime Application*
by Tim Monroe



The RezWack settings panel project window page 44

SECTION 7

- 22 **Process Control**
ps, top, kill
by Rich Morin

PROGRAMMER'S CHALLENGE

- 48 **AREA**
by Bob Boonstra

By Ilene Hoffman

USER SOLUTIONS TO PESKY TRASH PROBLEMS

How would you react if you tossed a bag of trash in the dumpster only to have it bounce back to your feet? Odd as this sounds, the inability to empty the Trash has been one of the more unexpected errors encountered all too frequently in Mac OS X. Web pages are filled with shareware solutions and instructions on how to drop into the command line interface to solve trash problems, but assuming you bought your Mac because you like the graphic user interface, let's look at some common problems and solutions to solving those annoying "can't empty the trash" messages.

First, let me caution you, the Trash Can should not be used to save files for later use. The Trash's function is to remove files from your drive. In fact, files aren't really deleted, but the pointers to the file (sort of like location coordinates) are removed, so that the hard drive space is made available for another file. This is why you can recover deleted files. That aside, when you put something in the Trash, you should delete it before shutting your computer down. Each user in Mac OS X has a separate Trash folder inside the Home folder. This folder is invisible to the user, but you can see files you put in the Trash until it is emptied.

If you are not sure you want to toss something, make a new folder inside your documents folder and call it Undecided, so that you can save it for later. You can drag the folder to the Dock, which puts an alias in your Dock for easy access.

To avoid the "Are you sure you want to remove the items in the Trash permanently" dialog, you can press Control and click on the Trash. Select this menu item to empty the Trash without the dialog. You can also press shift-option-command and the delete key to quickly remove a file. Some common errors and solutions are below.

PROBLEM: ERROR MESSAGE:

"The operation cannot be completed because you do not have sufficient privileges for 501" or "The operation cannot be completed because you do not have sufficient privileges for some of the items."

Explanation:

501 is the numeric equivalent of the first user set up in Mac OS X; this means you in most cases.

Privileges refers to who owns the file, who can write to it and who can read it, and therefore, who has the correct permissions to throw the file out.

Possible Causes and Solutions:

1. This may be caused by one of programs you've run incorrectly setting or altering the permissions on one or more of your files. An interruption in power, or an application freeze can also affect privileges.

You can check and/ or change the permissions of the file. Click on the file once and choose Show Info from the File menu (or press Command and I together). In the pop-up menu, in the Info window, choose Privileges and check the file's permissions. You need Read & Write permissions to delete a file. If a folder is the problem, remember to apply the changes to enclosed folders also. If you can't change the file's privileges, you can use Apple's Repair Privileges Utility to restore Apple-installed programs to their default permissions. If you are using Mac OS X 10.1.5. (You can find the Repair Privileges Utility at: <http://docs.info.apple.com/article.html?artnum=106900>.)

If these solutions don't work, you can also start up from Mac OS 9, use Sherlock to find the file and delete it. You cannot delete a file from OS 9 while in Classic Mode though.

2. Attempting to throw away a locked file can also generate the insufficient privileges message.

Again, choose Show Info from the File menu. Unclick the Locked check box and move the file into the Trash again to delete it. Prior to Mac OS X 10.1.5, if a file was locked while in Mac OS 9, you would have to restart in OS 9 to unlock the file before you could toss it. This seems to have been fixed in 10.1.5.

Ilene Hoffman is the Community Director and Editor of MacFixIt Forums and a Contributing Editor for MacTech magazine. She stole her first Mac from her Dad in 1984 after asking him to buy a PC. She's not touched a PC since! She is also the mother of a budding jazz musician and two capricious dogs.

NOW RUNNING ON YOUR MAC!

PROBLEM ERROR MESSAGE:

Can't empty the trash because ".trash items are in use."

Explanation:

Your computer thinks that a file is still open, and therefore cannot delete the file. Some of these problems may be fixed by upgrading to Mac OS X 10.1.5 or Mac OS X 10.2.

POSSIBLE CAUSES AND SOLUTIONS:

1. Check to make sure that the file is closed. Switch to the application used to create the file (click on the icon in the Dock). If you're not sure, quit the application used to create the file, then Empty the Trash.

2. If you are in the Columns View and the file is "open" in the Preview screen, the system is using the file. Put the file away by (i.e. take it out of the trash). Close the Preview screen in the Column View and drag the file to the trash again.

3. Some users have reported that waiting a while, and even tossing out another file and then emptying the Trash resolves this problem also.

If you want to resolve your trash problems by using other software, you can look at programs such as:

BatChmod which changes file permissions and can empty the Trash, or

Trash It! which is an AppleScript for emptying the Trash, or

ToTheTrash which helps you manage your about to be trashed files.

You can search Version Tracker (www.versiontracker.com) for other current Mac OS X utilities. You really shouldn't have to resort to using Terminal to empty your Trash.

Now, if you like typing, there are also solutions to fixing your Trash problems from the command line interface. In our next article we'll explore some of the those options for the geek in all of us.



IP*Works! The Only Truly Comprehensive Internet Toolkit

- More Than 30 Components Cover All Major Internet Protocols
- Follows Exact RFC Specifications
- Market-Tested For Over 7 Years
- In Use By Almost All Fortune 500s
- Royalty-Free Pricing

*IP*Works! for Mac OS X (10.0/10.1/10.2) gives you access to a host of new capabilities that will connect your applications to the Internet with unprecedented ease of use, power, and flexibility! You will be able to easily and quickly:*

- program the web: HTTP, WebForm, WebUpload
- call web services: SOAP, XMLp
- transfer files: FTP, TFTP
- send email: SMTP, FileMailer, HTMLMailer
- receive email: POP, IMAP
- read/post news: NNTP
- encode/decode: MIME, NetCode
- access directories: LDAP
- manage networks: SNMP, Whois, Ping, TraceRoute
- build clients and servers: IPPort, IPDaemon
- build packet applications: UDPPort, Multicast
- remote access: Telnet, Rexec, Rshell, RCP

...and a whole lot more! - download your free trial today from www.nsoftware.com!

IP*Works! for MacOS X is currently available as a C/C++ Library (OS X Framework). Objective-C Classes for Cocoa and RealBasic plugins coming soon at www.nsoftware.com

STAY TUNED!

By Andrew C. Stone

When Wild Animals Bite

Or How To Workaround Cocoa bugs

Of the last 50 articles I have written about Cocoa and Mac OS X, I think 49 have extolled the virtues of object oriented programming and the unparalleled advantages of using the dynamic runtime of Objective C. You might think I'd been fed some KoolAid, and you might not be too far off. But effects and mileage will vary.

The question that might arise in your mind is "Aren't there any serious drawbacks to using such a high level system?". And the answer to that would be the same as the drawbacks to any complex system: "Yes - Beware of Bit Rot", the inevitable complexification of simple architectures over time as more people work on it.

Once upon a time, the entire Cocoa development system was in the hands of a few able engineers, which kept it focused and robust between system releases. However, with Mac 10.2, aka Jaguar, we've seen a new and dangerous trend. Components that have been working almost flawlessly for over ten years have been "re-plumbed" without enough real world testing, and this can cause trouble for applications which push the envelope. Sometimes the pressure to release a new version is greater than the ability to do adequate quality control, so all of this is quite forgivable.

The same features that seem like an advantage can turn into a disadvantage! Since a Cocoa application links against the runtime AppKit and Foundation frameworks, when improvements are made to a subsystem framework, your application, even uncompiled, will take advantage of these improvements. For example, Mac OS X 10.2's text system introduced 3 new tab types: decimal, right aligned, and centered tabs. Users of our page layout and web authoring program, Create®, now automatically have access to these features under 10.2 - even with versions compiled under 10.1. By the same token, some changes in the subsystem framework can break existing, uncompiled applications.

Because of the relatively small number of Cocoa engineers at Apple compared to the number of traditional Carbon

engineers, there has been an increasing tendency to "pollute" our Cocoa purity with underlying Carbon implementations, and consequently, unexpected results can ensue! In Object Oriented theory, all objects are so well encapsulated and isolated that one can just swap out implementations of components and everything should just work. Reality just doesn't conform to these simplistic expectations.

WHEN CARBON MET COCOA

In the last several years of shipping and maintaining OS X applications, I've found that most of the problems have come up where traditional Mac OS 9 technologies have been shoehorned into the Cocoa model. The interstice between them is riddled with the same ambiguities that bedevils any organization that is a hybrid of philosophies and techniques. Cocoa's AppleScript implementation is an example of this - from the outside, it looks neat and clean, but getting it to work just right is a lot tougher than normal Cocoa programming tasks.

One object whose plumbing was changed in 10.2 is the `NSPrintInfo` - an object which maintains information about page size, orientation, margins and selected printer. The backend Print Manager grew out of the Carbon Tioga effort and is coded in C and C++. In 10.1, you could set the page size to any value (ideal for custom page sizes) with the simple invocation:

```
[myPrintInfo setPaperSize:NSMakeSize(someWidthInPoints,
someHeightInPoints)];
```

And the `printInfo` dutifully obeyed. This is useful for designing large scale posters to be printed offsite or for web sites with long content. However, in Jaguar, new behavior was introduced which attempts to see if that size is "valid" for the given printer. To be fair to the Apple engineers, I can see how this might be useful. But from my point of view, it broke the existing version of Create®! Luckily, since our corporate philosophy is free upgrades downloadable via the Internet, some quick coding, testing and uploading would solve the problem.

This new validation behavior occurs whenever "setPaperSize:" is called, and in the case of Create®, that's when the document is unarchived from a file. So, when you open an

Andrew Stone, founder of Stone Design <www.stone.com>, spends too much time coding and not enough time gardening.

Your Data Isn't an Important Part of the Job — It *IS* the Job.

VXA® FireWire

The High-Performance Tape Storage Solution For Apple Users

*"I can stick
my entire
hard drive
onto a tape
7 times
—that's just
plain cool!"*

Other backup media, such as CD-RWs and DVD-RAMs, simply can't compare to VXA-I tape technology when it comes to capacity, transfer rate and overall price.

Fast, economical and easy-to-use, VXA FireWire offers:

Ultimate File Security

- 100% Data Restore and Interchange
- Plug-and-Play Connectivity

Sharable Media

- Multi-Gigabyte File Transport
- Portable, Hot-Pluggable

Real Time Digital Video

Cross-Platform Compatibility

- FireWire/IEEE 1394/iLink Supported By All Major Manufacturers



Technology	Capacity in MBs	Transfer Rate in MB/sec	Price per MB
VXA Tape	33000	3	\$0.03
Imation Travan	10000	1	\$0.05
DVD RAM	9400	2.7	\$0.05
CD RW	700	1.9	\$0.50
Zip	250	1.2	\$0.76

LOWEST
COST per MB
OF ANY
Technology

Call Exabyte sales at 1-800-774-7172
or visit us on the web at www.exabyte.com

VXA® Tape Storage By **Exabyte®**



old Create document with a custom size, the old size is lost forever, and all of a sudden, size "A4" is chosen! What's worse is that any attempt to even read this paper size, such as with the public NSPrintInfo API call `-(NSSize)paperSize` or even `-objectForKey` on the dictionary returned from `-(NSMutableDictionary)dictionary` will also trigger this new validation behavior.

REDEMPTION SONG

So, what's the workaround? Somehow, we'll find Cocoa's redemption! For any flaw that comes in a shipping OS, there are ALWAYS tricky ways in Cocoa to do post-ship fixes! Once again, Objective C Categories save us from getting egg on our digital faces.

By examining the header file NSPrintInfo.h, we see a private instance variable (IVAR), `_attributes`, which is the actual mutable dictionary which holds all the values of the PrintInfo:

```
@interface NSPrintInfo : NSObject<NSCopying, NSCoding> {
    @private
    NSMutableDictionary *_attributes;
    void *_moreVars;
}
```

Because the IVAR is designated private, even a subclass cannot access it directly, so subclassing NSPrintInfo won't work. Only a category of the class containing the private IVAR can directly access the variable. Here's a category that can return any set value for `paperSize`, without triggering the unwanted validation behavior:

```
@interface NSPrintInfo (peek)
- (NSSize)grabOriginalSize;
@end

@implementation NSPrintInfo(peek)
- (NSSize)grabOriginalSize {
    id obj = [_attributes objectForKey:NSPrintPaperSize];
    return obj? [obj sizeValue]: NSZeroSize;
}

@end
```

Documents which had no custom page size will return `NSZeroSize`, which indicates that no extra work will be required. So, here's how we'll use the new category method:

```
NSPrintInfo *printInfo = [NSUnarchiver
unarchiveObjectWithData:obj];
// a freshly unarchived PrintInfo still has its old values in the dictionary...
if (printInfo) {
    NSSize raw = [printInfo grabOriginalSize];
    if (!NSEqualSizes(raw, NSZeroSize)) {
        // it could have been written
        // by a 10.1 version of Create
        [self setUpPrintInfo:printInfo toPaperSize:raw];
    }
    [self setPrintInfo:printInfo];
}
```

Now, all we need to do is fake NSPrintInfo out so that it thinks the custom paper size is valid. To do this, I had to dig into undocumented API using class-dump as explained in

several of my last few articles, and even more dreaded by an Objective C coder, into the Carbon header files! It turns out that there is a C function to make custom paper sizes programmatically, which stands to reason because the new Page Setup... panel in Jaguar has a popup option to set custom paper sizes:

```
OSStatus PMPaperCreate(PMPrinter printer, CFStringRef id,
CFStringRef name, double width, double height, const
PMPaperMargins *margins, PMPaper *paperP);
```

And we'll call it like this:

```
OSStatus osStatus = PMPaperCreate([[pi printer]
_printer], (CFStringRef)[NSString
stringWithFormat:@"%ld",self], (CFStringRef)[NSString
stringWithFormat:@"%2.2f X %2.2f",userW,userH],(double)
paperSize.width,(double) paperSize.height, marginPtr,
&myPaper);
```

If `PMPaperCreate()` returns an `OSStatus` of 0, then a new `PMPaper *` object (which you'll need to later release after use) has been created in `myPaper`. The first parameter, `PMPrinter`, can be returned from the `NSPrintInfo` via undocumented `NSPrintInfo` API, `-(PMPrinter)_printer`:

```
[[pi printer] _printer]
```

Another sweet feature of Cocoa is "toll-free bridging" between Core Foundation objects, such as a `CFStringRef` and the equivalent Cocoa object, in this case, the `NSString`. We will cast the `NSString` parameters to `CFStringRef`'s just to hush the compiler. The other fancy dancing that we're doing is using the pointer to memory of the document object as our unifying strategy for the second argument, but it could be any string as long as it is a unique identifier:

```
(CFStringRef)[NSString stringWithFormat:@"%ld",self]
```

Finally, we'll name the custom page in the user's units by converting the points to their measurement units - these functions are included below.

// these functions are declared here:

```
#import <CoreServices/CoreServices.h>
#import <ApplicationServices/ApplicationServices.h>

typedef struct OpaquePMPaper *PMPaper;
typedef struct {
    double top;
    double left;
    double bottom;
    double right;
} PMPaperMargins;
```

```
OSStatus PMPaperCreate(PMPrinter printer, CFStringRef id,
CFStringRef name, double width, double height, const
PMPaperMargins *margins, PMPaper *paperP);
```

```
- (void)setUpPrintInfo:(NSPrintInfo *)pi
```

```

toPaperSize:(NSSize)paperSize {
    if (floor(NSAppKitVersionNumber) <=
NSAppKitVersionNumber10_1) {
        /* On a 10.1 - 10.1.x system */
        // nothing special to do...
    } else {
        /* 10.2 or later system */
        float userW = convertToUserUnits(paperSize.width);
        float userH = convertToUserUnits(paperSize.height);
        PMPaperMargins margins;
        PMPaperMargins *marginPtr = &margins;
        // does C suck or what?
        PMPaper *myPaper;
        // Create uses WYSIWYG full page model:
        margins.top = margins.left = margins.bottom =
margins.right = 0.0;

        OSStatus osStatus = PMPaperCreate([[pi printer]
_printer], (CFStringRef)[NSString
stringWithFormat:@"%ld",self], (CFStringRef)[NSString
stringWithFormat:@"%2.2f X %2.2f",userW,userH],(double)
paperSize.width,(double) paperSize.height, marginPtr,
&myPaper);
        if (osStatus != 0) {
            // You may want to do something else here:
            NSLog(@"Trouble creating a custom page size: %f by
%f",paperSize.width, paperSize.height);
        }
        // on 10.1 this just works:
        [pi setPaperSize:paperSize];
    }

    // getting user units from an NSUserDefaults - sometimes C is cool!

    float
pointsFromUserUnits()
    {
        switch([[NSUserDefaults standardUserDefaults]
integerForKey:@"MeasurementUnits"]) {
            case 0: return 72.0;
            case 1: return 28.35;
            case 2: return 1.0;
            case 3: return 12.0;
            default: return 72.0;
        }
    }

    float
convertToUserUnits(float points)
    {
        return points/pointsFromUserUnits();
    }
}

```

CONCLUSION

So now, our printInfo will return the correct new size! Well, the dust hasn't settled entirely, so hopefully this will work and continue to work in the future. Ideally, NSPrintInfo would just "do the right thing" in terms of creating these custom page sizes. The proposed solution doesn't address custom page size unquing and coalescing of similar-sized pages - but then, I haven't finished coding the solution entirely either! Even the mighty Cocoa has its compatibility weaknesses as it grows, but its native power can even pull the tractor out of the mud when it gets really stuck.

New PRIMEBASE 4.0 Connectivity

PHP REALbasic

PRIMEBASE SQL DATABASE SERVER CONNECTIVITY

- REALbasic Plugin
- PHP 4
- JDBC
- ODBC
- EOF
- Perl
- WebObjects
- PrimeBase Application Server
- Witango
- Lasso
- 4D Plugin
- Omnis Studio
- C-API libraries

**[get a developer key
free of charge]**

download now!
www.primebase.com

AVAILABLE ON THE MOST POPULAR PLATFORMS

- Completely cross-platform
- Full-text searching and indexing
- Mac OS classic & X
- Mac OS X Server
- Linux
- Solaris
- IBM AIX
- all Windows platforms

**Develop on a Mac and deploy without any
additional effort on any platform you want.**

 **PRIMEBASE**

SNAP Innovation GmbH
Altonaer Poststraße 9a
D-22767 Hamburg / Germany
www.primebase.com
e-mail: info@primebase.com
Fon: ++49 (40) 389 044-0
Fax: ++49 (40) 389 044-44

By Ed Voas, Apple Computer, Inc.

HIObjct: The Carbon Object Model

Learn about the new Toolbox object model introduced in Mac OS X 10.2

INTRODUCTION

Apple's Human Interface Toolbox (HIToolbox) has always been object-oriented. There were various objects, such as windows, controls, menus, etc. and those objects could be manipulated by APIs meant to deal with them (`ShowWindow`, et. al.). But there was no really good way to override standard controls or even derive a new custom control from another control. HIObjct is a new mechanism that allows you to subclass and override standard toolbox objects as well as treat those objects in a polymorphic way. This article explains all you need to know about this new world, and serves as a foundation for learning about the other HIToolbox technologies that are in Jaguar.

WHAT IS HIOBJCT?

HIObjct is Apple's new common object base class in Jaguar for the HIToolbox. It is the foundation for everything Apple does these days in Carbon — it is something that all Carbon developers should learn about and understand.

People long assumed Apple had a C++ hierarchy under the hood for things like controls, but in reality they never did. The move to HIObjct was something that came out of the move to a real C++ framework internally for basic object types. Under the skin, the Jaguar Toolbox is a wildly different beast than in previous releases. Most of this change comes from Apple's re-architecting to use HIObjct and HView (the new view system in Jaguar).

Essentially, an HIObjct is any type of object that can send and receive Carbon Events — windows, menus, controls, etc. — basically all the objects that had an event target anyway. HIObjct is an object model where an HIObjct is the object-oriented encapsulation of an event target, and the 'methods' you call to manipulate this object are implemented as Carbon Events.

To be exact, windows, views/controls, menus, the application object, toolbars, and toolbar items all derive from HIObjct. **Diagram 1** shows a portion of the current

HIObjct hierarchy. The purpose of doing this is to gain some form of polymorphic behavior when dealing with these objects. For example, if you obtain a reference to a `WindowRef`, you can safely call HIObjct routines on it, such as `HIObjctGetEventTarget`.

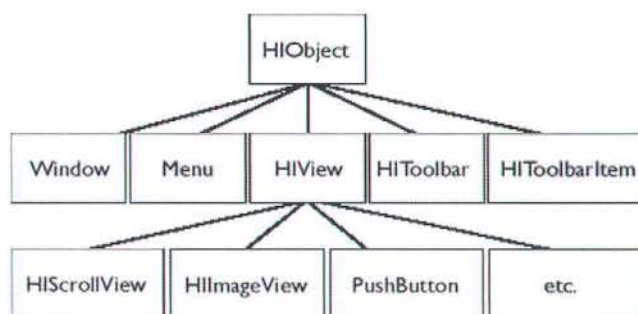
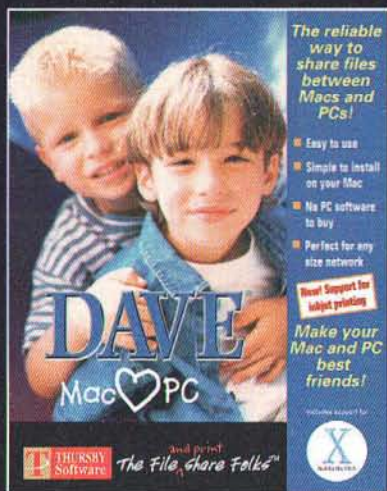


Diagram 1: The HIObjct Hierarchy

Having Toolbox objects all derive from a single-base class might not seem important at first glance, but as you start to use HIObjct more and more, you will realize that the Toolbox has just taken a huge step forward into an exciting new world. The more Apple improves the Toolbox, the faster they can make changes and add features. This change has done incredible amounts of good already — the addition of the Accessibility features in Jaguar required much less effort due to the new implementation.

The new data type Apple has introduced to represent an HIObjct is an `HIObjctRef`. The new objects introduced in the Jaguar Toolbox, such as toolbars, are merely typedef'd to `HIObjctRef`. Legacy types such as `ControlRef` are not typedef'd to maintain source code compatibility. For example, if Apple typedef'd `ControlRef` and `WindowRef` both to `HIObjctRef`, and you had an overloaded C++ method that took a `ControlRef` in one variant and a `WindowRef` in another, your code would probably no longer compile since they equated to the same type. Rather than wreak havoc, Apple decided to keep things the way they were. You can simply cast references to those types into `HIObjctRefs` as needed.

Ed Voas is the Manager/Tech Lead of the Carbon High Level Toolbox. When not coding, he is usually out applying 5 coats of polish to his car.



DAVE[®] SAVED MY MAC

They were going to take our Macs away. Then I met DAVE.

DINOSAUR FOUND IN OFFICE Manager Eaten Alive



“DAVE SAVED MY SANITY”

IT Analyst Timothy Hixon, Los Rios Community College System explained, “We didn’t have time to learn another network operating system. The Macs needed to fit seamlessly and efficiently into our district’s computer network, which used Windows NT file servers. DAVE was the answer.”

ALL THE BUZZ about DAVE

SHARE FILES AND PRINTERS - Mac/PC, PC/Mac

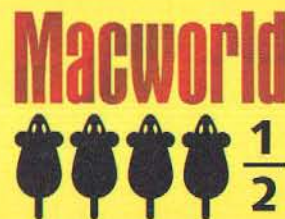
“DAVE is, without any doubt, the best solution for integrating your Mac into a PC environment.”

-Univers Macworld



“Essential Tool Gives Macs
Hassle-Free Access to
Windows Networks... this
product is a must-have.”

Shelly Brisbin
Macworld Magazine



How did DAVE save your Mac?

Write us and tell us your story of how DAVE solved your problem.

The entries chosen for publication will be awarded a special Thursby attache. Send entries to: DAVE-stories@thursby.com

www.thursby.com



and print
The File [^] Share Folks[™]

The new `HIObjct.h` header file contains a routine to create an `HIObjct`, but nothing to retain or release the object. Well then how are you supposed to use an object you could never destroy? Well, in reality Apple has done something cool — `HIObjcts` are actually Core Foundation types! That means you can add anything that is derived from `HIObjct` (even a window) into a CF collection, such as a `CFArray`. It also means you simply use `CFRetain` and `CFRelease` to retain/release the object. Wicked! But you do need to be careful about such things, as windows for example may not go away when you expect them to if you are retaining them in places. You can also cause circular retention (which sounds very painful), so be careful.

POLYMORPHIC FUNCTIONS

Let's first check out the routines you can call on any `HIObjct`. There aren't that many at present. That's a good thing, as it makes it easier to learn.

Event Handling

To get the event target of any `HIObjct`, you merely call `HIObjctGetEventTarget`. This is very nice! It means you can now keep an array of dissimilar objects such as controls and windows, and just iterate the list, getting their event targets. For example, you can use this technique to keep a list of objects that are interested in receiving particular notifications. It doesn't matter if it's a window, control, or toolbar item. All you care about is that they accept carbon events. There is no need to track what type of object they are and call the appropriate API to get the target.

Class Identity

An `HIObjct` class is uniquely identified by its class ID, which is a `CFString`. Apple uses Java-style namespacing for its classes (`com.apple.blah`). To obtain the class ID of an object, you call `HIObjctCopyClassID`. This returns a copy of the class ID string for you to inspect. You can compare it to other class IDs to see if an object is of a particular type, for example. Be warned that if a class ID is not documented in any header (and at present only two of them are), you should not rely on those class IDs remaining constant between releases of Mac OS.

It's also useful to find out if an object you have is something of a specific type — typically referred to as an 'is-a' test. For example, if you have a push button's `ControlRef` in hand, you can see if it's a control by asking if it is derived from `HIView`:

```
if ( HIObjctIsOfClass( anObjct, kHIViewClassID ) )
{
    // It's an HIView!
}
else
{
    abort();
}
```

Believe it or not, we're running out of polymorphic functions! As I mentioned before, you can use CF routines to retain or release an `HIObjct`. So the following is perfectly legal now:

```
CreatePushButtonControl( window, ..., &control );
```

```
// do some fun stuff here, maybe add it to an array,
// which will retain it. Removing it from the array
// would decrease the retain count as expected. You
// must use the standard CF type callbacks when you
// create the array though.
```

```
CFRelease( control );
```

`CFRelease` is now a synonym for `DisposeControl`, `DisposeWindow`, and `DisposeMenu`. New types (like the toolbar) have no specific retain or release calls of their own.

Debugging

In the Toolbox, there are several functions you can call to print debugging info for an object to `stdout`. Some of them aren't exported, so you can only call them from `gdb`. And the names are not consistent — even engineers at Apple can seldom remember what they are. This was fixed in `HIObjct` by introducing one base class function to display the debugging information for an object — `HIObjctPrintDebugInfo`. Call it with a window or control reference and you will see all the types of information it prints to `stdout`.

And much much more!

Well, not really. There are a couple of routines to deal with Accessibility, another major feature in Jaguar. Accessibility is well outside the scope of this article though, and deserves an entire article of its own.

There are also a couple of other APIs you should know. These are important primarily when creating objects of your own design. That's precisely what we'll cover next, so we'll talk about them as we go.

CREATING YOUR OWN OBJECTS

You can create `HIObjct` classes of your own and even subclass ones provided by the Toolbox. You would most likely do this to create a custom view or toolbar item. Or you might create a custom `HIObjct` so you can have your own event target to pass to Toolbox APIs.

Let's get right into how to subclass something. We will create a simple subclass of `HIObjct`. The first thing that we need to do is register our new subclass with the `HIObjct` system. You do this via a call to `HIObjctRegisterSubclass`:

```
extern OSStatus
HIObjctRegisterSubclass(
    CFStringRef      inClassID,
    CFStringRef      inBaseClassID,
    OptionBits       inOptions,
    EventHandlerUPP   inConstructProc,
    UInt32           inNumEvents,
    const EventTypeSpec * inEventList,
    void *           inConstructData,
    HIObjctClassRef * outClassRef );
```

For our purposes, we would call this function as follows:

```
const EventTypeSpec kMyFunObjectEvents =
{ | kEventClassHIObjct, kHIObjctConstruct |,
  | kEventClassHIObjct, kHIObjctDestruct |
};
```

DISCOVERY AHEAD

EXPLORE, EXPERIENCE, DECIDE.

Your Future is at COMDEX.

Keynotes—Free to all COMDEX attendees

COMDEX Fall 2002 is your best opportunity to experience, explore, and learn about the latest product innovations, leading vendors, and powerful new business strategies.

- FREE access to the exhibit floor
- FREE entrance to the Great Debates and Hot Spots
- FREE admission to the new Digital Lifestyles Conference and SuperSession
- Content-rich Educational Programs for business and technical professionals

Register today!

Visit www.comdex.com/fall to register or call 888-568-7510.
Use Priority Code MTAS and Coupon Code 342.

COMDEX EDUCATIONAL PROGRAMS: NOVEMBER 16–21, 2002

COMDEX EXHIBITION: NOVEMBER 18–22, 2002

LAS VEGAS CONVENTION CENTER, LAS VEGAS, NEVADA



BILL GATES
Microsoft Corporation



HECTOR DE J. RUIZ
AMD



CARLY FIORINA
Hewlett-Packard
Company



PETER CHERNIN
News Corporation and
Fox Group



SCOTT MCNEALY
Sun Microsystems, Inc.



BRIAN HALLA
National Semiconductor
Corporation

OFFICIAL CORPORATE SPONSORS OF KEY3MEDIA GROUP



Official Card of COMDEX



Official Automotive Sponsor



New York Stock Exchange



PRESENTS

COMDEX®
FALL 2002

THE GLOBAL TECHNOLOGY MARKETPLACE

Copyright © 2002 Key3 Media Events, Inc., 5100 Wilshire Boulevard, Suite 325, Los Angeles, CA 90048-3659. All Rights Reserved. COMDEX 2002, Key3 Media, COMDEX, and associated design marks and logos are trademarks owned or used under license by Key3 Media Events, Inc., and may be registered in the United States and other countries. Other names mentioned may be trademarks of their respective owners.

```

#define kMyFunObjectID \
CFSTR( "com.mycompany.funobject" )

HIObjRegisterSubclass(
    kMyFunObjectID,
    NULL, // no base class
    0, // no options
    MyHandler,
    GetEventCount( kMyFunObjectEvents ),
    kMyFunObjectEvents,
    0, // no handler data
    &classRef );

```

By passing NULL for the `inBaseClass` parameter, we are telling the function that we want to be a subclass of `HIObj` itself. This function sets up `MyHandler` as a handler that will automatically get pushed onto an instance of this class when one is created. In this example, this handler only deals with two events — construct, and destruct. These two events are, in fact, required for any subclass you are registering. If you try to register a class handler that does not respond to these events, the earth will open up and swallow you. Either that, or an error will be returned.

Figure 1 shows our class handler for our custom object.

Listing 1: Class event handler

```

// simple object
struct MyFunObject {
    HIObjRef ref;
    Boolean isFunEnabled;
};

OSStatus MyHandler(
    EventHandlerCallRef inCallRef,
    EventRef inEvent,
    void * inUserData )
{
    OSStatus result = eventNotHandledErr;
    UInt32 theClass, theKind;
    MyFunObject* object = (MyFunObject*)inUserData;

    // Please note that object above is overloaded in this handler.
    // for the kEventHIObjConstruct event ONLY, the inUserData
    // parameter is the user data you passed to HIObjRegisterSubclass.
    // For all other calls to this function, it is the object pointer that you create
    // and return in your handling of the kEventHIObjConstruct as seen below.

    theClass = GetEventClass( inEvent );
    theKind = GetEventKind( inEvent );

    switch( theClass )
    {
        case kEventClassHIObj:
            switch( theKind )
            {
                case kEventHIObjConstruct:
                    {
                        HIObjRef ref;

                        // When the construct event is called, you are handed the
                        // HIObjRef that is being constructed. In this example,
                        // we save it off in our object. This is what you generally
                        // want to do so you can call appropriate Toolbox APIs
                        // as needed, since your object pointer (created below),
                        // is not a real HIObj.

                        result = GetEventParameter( inEvent,
                                                    kEventParamHIObjInstance,
                                                    typeHIObjRef,
                                                    NULL,
                                                    sizeof( HIObjRef ),
                                                    NULL,
                                                    &ref );

                        require_noerr( result, ParameterMissing );

                        // Create the fun object here. If we fail to malloc it, return

```

```

// an error. (If you ever wondered why sometimes the Toolbox
// returns memFullErr on a system where you can never run
// out of memory without the system dying a horrible death,
// now you know!) We are reusing the object variable above
// since it's not used for anything in this particular example
// during construction.

```

```

object = malloc( sizeof( MyFunObject ) );
require_action( object, CantAllocObject,
                result = memFullErr );

```

```

object->ref = ref;
object->isFunEnabled = false;

```

```

// OK. Here's the key: we replace the instance parameter
// with our object pointer. The type of the parameter MUST
// be typeVoidPtr. It's the Law. The Toolbox will store this
// off with the HIObj. This will allow you to call
// HIObjDynamicCast later if you need to get your
// object pointer back from an HIObjRef.

```

```

SetEventParameter( inEvent,
                    kEventParamHIObjInstance,
                    typeVoidPtr,
                    sizeof( void * ),
                    &object );

```

```

}
break;

```

```

case kEventHIObjDestruct:
    // This is easy. Just dispose of the object. Do NOT call through
    // with CallNextEventHandler — Very Bad Things will happen.
    // This is a top down destruction. Don't try to get fancy!

```

```

free( object );

```

```

break;

```

```

}
break;

```

```

CantAllocObject:
ParameterMissing:

    return result;
}

```

How an HIObj is constructed

In order to truly understand what is going on in the class handler, let's discuss the steps the Toolbox takes when creating an `HIObj`.

The Toolbox sends construction events bottom-up, as you would expect in C++ or similar runtime models. This means that base classes are constructed before subclasses.

First, the Toolbox creates the base `HIObj`. This is where the actual `HIObjRef` value is created. It is important to note that unlike C++ (where the subclass' *this* pointer is the same value as the base class' *this* pointer), a subclass' object pointer is not technically an `HIObj`. It is merely data stored with the `HIObj` for the specific class. We'll see how this works later. After the base `HIObj` is created, all other subclasses of `HIObj` between it and your class are constructed. This means that if you are creating a object of type `Foo`, which derives from `Bar`, which derives from `HIObj`, first the `HIObj` is created, then `Bar`, and finally comes your 15 minutes of stardom.

Once the Toolbox creates your immediate superclass, it starts the process of constructing your part of this aggregate `HIObj`. First, it takes the event handler you registered and installs it onto the event target that was created for the `HIObj`. As mentioned, this handler must be registered for the `kEventHIObjConstruct` and `kEventHIObjDestruct` events.

Special Small Dogs

Here are just a few of the photos our customers have sent us of their special dog friends!



Check out all the other special Small Dog photos and send us one of your own at www.smalldog.com!

Small Dog's Special



PowerBook G4/667 Titanium
\$2499

- 512MB RAM
- 30gb Hard Drive
- Combo Drive
- Airport card and base station



Purchase from an Apple Specialist!

Small Dog Electronics is proud to have earned Apple's prestigious designation of Apple Specialist. Small Dog Electronics has exceeded all of Apple's stringent requirements for its Specialists and has taken it one step further.

Small Dog sells only the most powerful personal computers in the world—every single one of them an Apple Macintosh! In addition to the rigorous Apple training program, each Small Dog employee has chosen a specific area of expertise to concentrate upon. You can be assured that whomever you talk to at Small Dog, from our sales engineers to our shipping department, you are talking to a trained, enthusiastic, Apple Macintosh Specialist!

Small Dog Electronics is also an Authorized Apple Service Provider Plus. We have certified Apple Technicians that utilize genuine Apple parts for all repairs and provide technical support for our customers based upon their experience in upgrading and supporting thousands of Apple Macintosh computers for our customers!

Talk To an Apple Professional!

Did you know that when you call Small Dog Electronics, you are most likely talking to an Apple Product Professional? Each year, we participate in Apple's on-line courses and seminar training programs to learn as much as we can about the products that we sell and service!

Exclusive Top Dog Membership Treats

For each dollar purchased on-line at the Small Dog web site, you will receive a doggie treat. You can redeem your accumulated treats for Small Dog or Apple apparel and other products. You are automatically enrolled and begin accumulating treats in the Top Dog Club with your first purchase. Remember the more you buy, the more treats for you!



**Small Dog
Electronics**

1673 Main Street
Waitsfield, VT 05673 USA
Phone: 802-496-7171
Online: smalldog.com



Next, the Toolbox *directly* calls your handler with a `kEventHIObjConstruct` event. When called directly, you are not being called in the context of a handler stack, so you cannot call `CallNextEventHandler`, unless you like to crash. The `inUserData` parameter of your class handler is passed the value you specified for the `inConstructData` parameter when you registered the class. Typically, during construction you will allocate memory for your own instance data. This allocation might be as simple as calling `malloc` or `NewPtr`, or it might involve creating your own C++ object. In the construct event, you are passed the base `HIObjRef` of the object being created. You should store this `HIObjRef` in your own instance data for later use. You should then use `SetEventParameter` to set the `kEventParamHIObjInstance` parameter in the construction event with your instance data. You must use `typeVoidPtr` as the type.

Once back from sending the event, the Toolbox looks for your instance of `typeVoidPtr` in the event and stores it with the object. It also sets the user data parameter of the event handler it installed to be this instance data. Following the construct event, all calls to your event handler will have the instance data you returned to the Toolbox. At this point, all events are now sent to your object using standard Carbon Event mechanisms. It is only the construct event that is special.

Once construction has completed successfully, the Toolbox will send your object a `kEventHIObjInitialize` event. The initialization stage is optional; i.e. an object does not need to respond to the initialize event unless it is expecting certain parameters to be passed to it at creation time. This is where those parameters may be fetched. We'll show an example of this shortly. The first thing you should do is call through to the 'inherited' method with `CallNextEventHandler`. Once back from that, you should verify the result code returned is `noErr`, indicating that the base class initialized properly. If it did, you should extract any initialization parameters and do whatever your object requires in order to properly initialize. If the base class did not initialize properly, you should return the error that `CallNextEventHandler` returned as the result of your handler immediately, doing no work. The Toolbox will see the error code and proceed to destroy the object (see 'Object Destruction,' below). Your object must be able to be destroyed in a partially initialized state such as this.

Upon successful initialization, the `HIObjRef` is returned to the caller of `HIObjCreate`. From there, you can have all sorts of cool fun.

Object Destruction

Destruction is top down, as in C++. When an object's retain count reaches zero, the object is destroyed. During destruction, the Toolbox sends a `kEventHIObjDestruct` event to the object. This event will just propagate using the normal rules of event handlers (top-down), which is exactly what we want. It is a *very* bad thing to call `CallNextEventHandler` during destruction. Just clean up and return from your handler.

Creating an instance of your class

To create an instance of this new object class, all we need to do is make this call:

```
HIObjRef obj;
```

```
err = HIObjCreate(
    kMyFunObjectID,
    NULL, // no initialize event
    &obj );
```

At this point, you have a nice object of your own design. But it doesn't do much, does it? You can create it and release it. Let's add an API to set the fun enabled boolean. See **listing 2** for the code.

Listing 2: Adding APIs to your class

```
OSStatus MyFunObjectSetFunEnabled(
    HIObjRef inObj,
    Boolean inEnabled )
{
    MyFunObject* obj;
    OSStatus err = noErr;

    // Cast the HIObjRef handed to us to our internal instance data.
    // What this does is look up the data we returned in the
    // kEventParamHIObjInstance parameter when we handled the
    // kEventHIObjConstruct event in listing 1. If this function
    // returns NULL, then the object is not of the class specified in
    // the second parameter.

    obj = (MyFunObject*)HIObjDynamicCast(
        inObj, kMyFunObjectID );

    require_action( obj, InvalidObject,
        err = kMyInvalidClassID );

    // OK. We have our object now. Store the value

    obj->isFunEnabled = inEnabled;

InvalidObject:
    return err;
}
```

This is pretty straightforward, but there is one oddity — the dynamic cast call. It's necessary because the API we wrote takes an `HIObjRef` and not a `MyFunObject` pointer. This starts getting into the ugly truth of it all — your objects are not really `HIObj`s! When it comes right down to it, they are just some value that you wish to associate with an `HIObj`.

Diagram 1 shows a comparison between a C++ object layout and an `HIObj` layout. In C++ the object is a unified block of memory. It can do this because it's part of the language runtime. With `HIObj`s, the object reference always points to the `HIObj`, and the data stored by subclasses are kept track of in the `HIObj` itself.

Remember that when you handle the `kEventHIObjConstruct` event, you are given the `HIObjRef` for your object ahead of time. The object we created in **listing 1** is just our blob of data that we want stored with the `HIObj` for our class. We get that data back with `HIObjDynamicCast` if we happen to have an `HIObjRef` in hand.

If you think about it though, this 'casting' mimicks C++ very well, in that you can't take a pointer to a base class in a class method without 'casting up' to your class before using it. So while the guts are different from C++, the mentality is not. The main difference is that we can't take our struct pointer and treat it exactly like an `HIObj`, because, like, it's not.

One last thing to note is that while HIObjecT's data layout is somewhat disjoint, the 'vtable' layout is just as you'd expect. There is one unified Event Target that acts as the vtable. Only the data is spread out. Also, there's nothing to say that in the future Apple couldn't come up with some superior scheme to try to make it better. There is an 'options' parameter in the call to HIObjecTRegisterSubclass, so it would be possible to define new types of subclasses if Apple so desired.

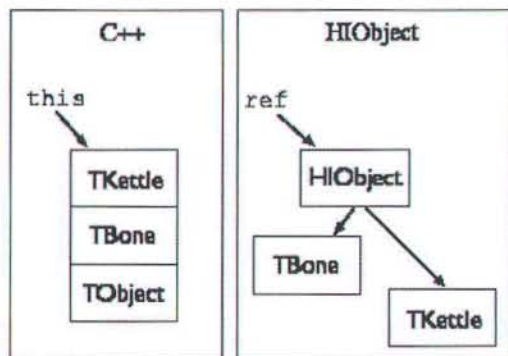


Diagram 1: C++ vs. HIObjecT Data Layout

At this point, you know all the hard stuff. The rest is just academic. Let's extend our object a bit. First, let's write a new API to create an object. It will wrap the call to HIObjecTCreate. It will also take a parameter indicating the initial value of our 'fun enabled' boolean.

Listing 3: Our Creation API

```
OSStatus MyFunObjectCreate(
    Boolean    inFunEnabled,
    HIObjecTRef* outObject )
{
    EventRef    event;
    OSStatus    err = noErr;
    HIObjecTRefobjecT;

    //To pass parameters to our object at creation time, we need to use a
    // Carbon Event. We create it here and add our boolean parameter. We
    // then pass it into HIObjecTCreate. This event will be sent to our
    // object at initialize time. You must take care to use the correct class
    // and ID for this event.

    err = CreateEvent( NULL, kEventClassHIObjecT,
        kEventHIObjecTInitialize, GetCurrentEventTime(),
        0, &event );
    require_noErr( err, CantCreateInitEvent );

    SetEventParameter( event, kMyFunEnabledParam,
        typeBoolean, sizeof( Boolean ), &inFunEnabled );

    err = HIObjecTCreate( kMyFunObjectID, event, outObject );

    ReleaseEvent( event );

CantCreateInitEvent:
    return err;
}
```

To pass initial parameters to an object, you need to create a Carbon Event with the right class and kind. You simply add your parameters onto that event and pass it into HIObjecTCreate. The sort of code you see in **listing 3** is exactly the sort of stuff Apple does in the Toolbox. Calls such as

CreatePushButtonControl are implemented using this exact same formula.

Now, in our class handler, we need to add a case to handle the initialize event. See **listing 4** for how to do that.

Listing 4: Adding the Initialize Handler

```
OSStatus MyClassHandler(
    EventHandlerCallRef    inCallRef,
    EventRef               inEvent,
    void *                 inUserData )
{
    OSStatus    result = eventNotHandledErr;
    MyFunObject* object = (MyFunObject*)inUserData;

    .
    .
    .
    case kEventHIObjecTInitialize:
        // Be sure to call our 'inherited' initialize first. Then extract
        // our parameter and leave.

        result = CallNextEventHandler( inCallRef,
            inEvent );
        if ( result == noErr )
        {
            result = GetEventParameter( inEvent,
                kMyFunEnabledParam,
                typeBoolean, NULL,
                sizeof( Boolean ), NULL,
                &object->isFunEnabled );
        }
        break;

    .
    .
    .
}
```

high quality - competitive rates - 16 years experience - award winning

Full Spectrum Software

Development & Testing

Device Drivers

Porting

TCP/IP

Carbon / OSX

Plug-ins

Cross Platform Development

One Bridge Street
Newton, MA 02458

617-965-0029

www.FullSpectrumSoftware.com

competitive rates - 16 years experience - award winning - high quality

reliable - high quality - competitive rates - efficient - award winning - qa services - reputable

reliable - high quality - competitive rates - efficient - award winning - qa services - reputable

Well, that was easy. Of course, your initialize handler might be much more complicated. As seen in the code, you should call through before handling the initialize event yourself. There are some exceptions to this, but it depends on how your object is set up and whether initializing the base class will cause one of your event handlers to be called before you are ready. But you should treat calling through first as the rule.

Now, for completeness, let's add support for a couple of other things. First, let's add support for equality testing. If someone has two references to two of your object instances, they could call `CFEqual` on them to see if they are the same. `CFEqual` calls into the `HIObj` implementation, and that in turn sends a Carbon Event to your instance. By the time the event gets to you, the Toolbox has already checked to see whether the references are identical and that the class of object is the same. So you know that you will be passed an instance of the same class as yours. All you need to do is compare your internal state and return the result in the event. **Listing 5** shows the handler code to do this.

Listing 5: Adding the Equality Handler

```
OSStatus MyClassHandler(
    EventHandlerCallRef    inCallRef,
    EventRef               inEvent,
    void *                 inUserData )
{
    OSStatus    result = eventNotHandledErr;
    MyFunObject* object = (MyFunObject*)inUserData;

    .
    .
    .
    case kEventHIObjIsEqual:
    {
        Boolean    localResult;
        HIObjRef otherHIObj:
        MyFunObject* other;

        // Get the direct object. It will be the object we are being
        // compared to.
```

```
err = GetEventParameter( inEvent,
    kEventParamDirectObject,
    typeHIObjRef, NULL,
    sizeof( HIObjRef ), NULL,
    &otherHIObj);
require_noErr( err, MissingParameter );

// Now cast it to get the data pointer for the other object.
// This cast should never fail since we are guaranteed to
// be looking at an object of the same class by the time
// we get here.

other = (MyFunObject*)HIObjDynamicCast(
    otherHIObj, kMyFunObjID );
check( other != NULL );

// compare the two objects' guts. For our object types, we'll
// consider them equal if their isFunEnabled settings are the same.

localResult = ( other->isFunEnabled ==
    object->isFunEnabled );

// Now store the result in the kEventParamResult parameter
// as typeBoolean. We are done. Exit with an appropriate result.

SetEventParameter( inEvent,
    kEventParamResult,
    typeBoolean, sizeof( Boolean ),
    &localResult );
result = noErr;
}
break;

MissingParameter:
    return err;
}
```

As you can see, it's simple to handle the equality event. Just extract the direct object parameter and cast it to get the object data pointer. Then make the comparison however your class decides to and store the result in the event.

There's only one more event to handle: `kEventHIObjPrintDebugInfo`. This is sent to your handler when

eat.
sleep.
Learn
Cocoa®.



Big
nerd
ranch

Intensive Classes for Programmers
www.bignerdranch.com
404.210.5663

someone calls `HIObjPrintDebugInfo`. Big surprise, I'm sure. Check out **listing 6** for how to handle this event.

Listing 6: Adding the Debugging Handler

```
OSStatus MyClassHandler(
    EventHandlerCallRef    inCallRef,
    EventRef               inEvent,
    void *                 inUserData )
{
    OSStatus    result = eventNotHandledErr;
    MyFunObject* object = (MyFunObject*)inUserData;

    .
    .
    .
    case kEventHIObjPrintDebugInfo:
    {
        fprintf( stdout, "MyFunObject" );
        fprintf( stdout, "    Fun Enabled: %d",
            object->isFunEnabled );
        result = noErr;
    }
    break;

    .
    .
    .

MissingParameter:
    return err;
}
```

Wow. That was simple. All you do to respond to this is print your information to `stdout`. Typically, you'd print the name of the class, and then any information under that, indented a bit. The Toolbox has a standard formatter for its output. You should in general try to match the look of the Toolbox output. In the future, this formatter might be exposed for use by developers.

THAT'S ALL FOLKS

Well, believe it or not, you now know all you need to about HIObj. We've covered the different polymorphic functions and shown you how to create HIObj classes and objects of your own design. We've also demonstrated dynamic casting and covered every event that gets sent to your HIObj from the HIObj subsystem.

With this knowledge in hand, you can do things such as add custom toolbar items for the new `HIToolbar` class in Jaguar. You do this by subclassing the `HIToolbarItem` class using the steps shown in this article. You can also start to write custom views based on `HIView`. Remember, HIObj permeates everything in the Toolbox in Jaguar, so it's a good thing to understand what it is and how it works. Get out there and start creating your own HIObj's!

Special thanks to David McLeod, Eric Schlegel, Guy Fullerton, Curt Rothert, Matt Ackeret, and Bryan Prusba for reviewing this article.



Fetch

Top dog.

X

Fetchsoftworks.com

Version 4.0.2 now available.

By Rich Morin

Process Control

ps, top, kill, ...

A recent foray into Internet Radio conflicted in some manner with the OSX screen saver on my desktop machine. As a result, I was unable to get the screen saver to go away. I tried all of the usual tricks: hitting mouse buttons, moving the mouse, advancing to keyboard keys (e.g., shift, return) and finally "magic key combinations" (e.g., command-option-escape). No luck.

On Mac OS 9, my next steps would have been to try a "three-finger salute" (i.e., command-control-power), followed by pushing the reset button and if need be, pulling the plug. This being a BSD-based machine, however, I had a better solution at hand.

I went to another machine on the local net, logged into my desktop machine, and killed a couple of sorely-confused processes. I then logged out and returned to my desk, flushed with victory and the knowledge that I had a useful Section 7 topic in hand.

ADVANCE PREPARATION

My "remote login" trick wouldn't have worked without a bit of advance preparation. OSX, for very good reasons, turns off most "remote services", by default. I don't recommend turning services on, willy-nilly, but `ftp` and `ssh` are so handy that I wouldn't want to live without them. And, because my desktop machine is hidden behind a firewall, I consider the risk to be pretty minimal (your mileage may vary).

In any case, here's how to allow remote `ftp` and `ssh` on an OSX 10.2 machine: In the System Preferences application, select the Sharing dialog. Under the "Services" tab, turn on the checkboxes labelled "FTP Access" and "Remote Login".

You will also need a way to "find" your machine from the other one. This can be achieved in any number of ways, including DNS (Domain Name System), `/etc/hosts` files, etc. Your network administrator will (or should :-)) know how to make this happen on your local network. As a last resort, you can use your machine's IP (Internet Protocol) address as a name.

LOGGING IN

Because `ftp` and `ssh` aren't OSX-specific tools, you can log in remotely from almost any operating system. This being an OSX-

related column, however, I'll assume that you're running OSX. If you're not, you'll have to deal with the vagaries of installing and starting up the needed clients on the machine you're on. On OSX, just start up a Terminal window and type:

```
% ssh <me>@<mydesk>
<me>@<mydesk> password:
```

The first time you do this from a given machine, you'll be asked whether you trust the SSH keys on your desktop machine. Say yes (-:). After entering your password, you'll get a prompt from your desktop machine:

```
<me>@<mydesk> [~] 1:
```

There are all sorts of things you can do at this point, but try not to get too excited and do something you'll regret. Think "delicate surgery", rather than "hack and slash". Thus, the first thing we want to do is find out a bit about the programs that are running over on the desktop machine. To do this, we'll use the `top` command, which displays system usage statistics and then lists the top processes, in terms of CPU utilization:

```
<me>@<mydesk> [~] 2: top
Processes: 65 total, 3 running, 62 sleeping... 190 threads
17:39:16
Load Avg: 2.15, 1.46, 1.35
CPU usage: 62.3% user, 37.7% sys, 0.0% idle
SharedLibs: num = 116, resident = 23.2M code, ...
MemRegions: num = 4936, resident = 159M + 13.0M private, ...
VM: 4.22G + 45.8M 5363(0) pageins, 0(0) pageouts
```

PID	COMMAND	%CPU	TIME	...
758	top	7.9%	0:38.79	...
703	tcsh	0.0%	0:00.34	...
702	sshd	0.0%	0:00.78	...
701	ssh	0.0%	0:00.86	...
692	CCacheServ	0.0%	0:06.09	...
610	Finder	0.0%	0:18.75	...

Unlike most BSD commands, `top` doesn't just run and go away. Instead, it refreshes its display about once a second, letting you see if things are changing. When you're done viewing the display, type "q" (or Control-C) and the program will terminate. Alternatively, just start up another Terminal window, leaving `top` running in the first one...

The system usage statistics are a bit arcane; look at `top`'s man page if you want the details. For now, just note that the number of

Rich Morin has been using computers since 1970, Unix since 1983, and Mac-based Unix since 1986 (when he helped Apple create A/UX 1.0). When he isn't writing this column, Rich runs Prime Time Freeware (www.ptf.com), a publisher of books and CD-ROMs for the Free and Open Source software community. Feel free to write to Rich at rdm@ptf.com.

processes should normally be under 100 and that the load average (time-averaged system load) should be under five. If you see higher values on your machine, you may have a runaway application.

Because `top`'s process list is sorted by CPU utilization, active programs tend to "bubble up". So, look near the top of the list for suspects. In some cases, killing off a single process won't be enough; to restore my desktop to proper functionality, I had to kill the screen saver, the Internet Radio application, and the Finder!

I issued a separate `kill` command for each process I wanted to terminate, specifying the PID (process ID). This let me see and evaluate the effect of each action:

```
^C<me>@<mydesk> [~] 3: kill -9 610
```

Be sure to get the PID right; you don't want to kill off an innocent application, by mistake! By the way, `kill` can do more than just terminate processes. It can send any desired "signal" to one or more specified processes. So, for example, it can be used to tell processes to check their configuration files, etc.

If you can't figure out which command you need to kill, you may need to get a different view of the information. The `ps` command lists "process status" information, in a variety of formats. For our present purposes, "`ps awx`" is a reasonable idiom:

```
<me>@<mydesk> [~] 4: ps awx
PID  TT  STAT  TIME COMMAND
  1  ??  Sls   0:00.03 /sbin/init
  2  ??  SL    0:02.35 /sbin/mach_init
 41  ??  Ss    0:02.24 kextd
...
```

To list the commands in a different order, just pipe together some commands:

```
<me>@<mydesk> [~] 5: ps awx | sort +3 -rn | more
 62  ??  Ss   56:09.79 .../WindowServer
...
PID  TT  STAT  TIME COMMAND
994  std  R+    0:00.00 more
993  std  R+    0:00.01 sort +3 -rn
...
 610  ??  S     0:18.80 .../MacOS/Finder -psn_0_3538945
...
```

Transliterated, the pipeline above says to run the output of `ps` through a line-oriented `sort` (sorting by the fourth column, in reverse numeric order), then pipe the result through `more` (a text viewing command). If you know what command you're looking for, of course, you can use `grep`:

```
<me>@<mydesk> [~] 6: ps awx | grep Finder
1006 std  R+    0:00.00 grep Finder
 610  ??  S     0:18.80 .../Finder -psn_0_3538945
```

With the advent of OSX 10.2, I feel more comfortable in recommending FreeBSD-related books to OSX users and programmers. There are still some differences, to be sure, but any book that covers FreeBSD 4.X commands will be a good match for the OSX 10.2 command set.

Although processes are the workhorses of any BSD system, there are few books that spend much time on them. Quite a few man pages relate to processes, however, and the DOSSIER (www.ptf.com/dossier) volume "Processes: FreeBSD" brings them together in a single volume.

Impact your bottom line while working with dedicated proponents of Apple technologies

With us, your only constant is success. Our custom solutions help our clients excel their business objectives and effectively target their audience. That's why we suggest that we run your technology while you play the winning shot.

Core Competencies

- WebObjects development
- Mac OS X application development
- Cocoa application porting
- Carbon porting
- Mac OS product maintenance
- Windows/Mac OS/Unix porting
- Cross-platform development

Service Offerings

- Offshore Project Development
- On-Site Staff Augmentation
- Off-Site Project Development
- User Training

For More Information

visit <http://www.avestacs.com>
Email: mithu@avestacs.com

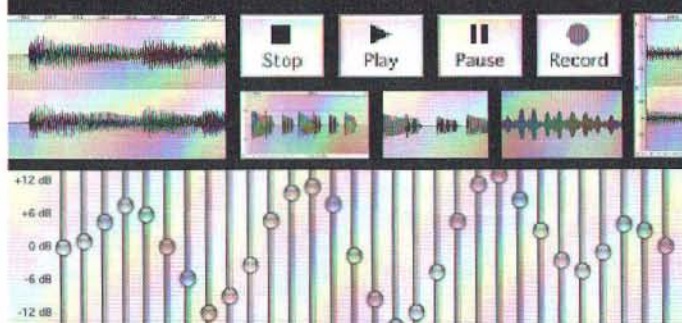


Avesta Computer Services, Ltd.
USA . EU . Asia



audio recording • editing • effects

Felt Tip Sound Studio 2.1



FELT TIP SOFTWARE presents a MAC OS X application "SOUND STUDIO"
available on CD-ROM and as a WEB DOWNLOAD
sales by KAGI written by LUCIUS KWOK
© 2002 Felt Tip Software. All rights reserved.



www.felttip.com/ss



by Tim Monroe

Human Resources

Adding Macintosh Resources to a Windows QuickTime Application

INTRODUCTION

Macintosh applications use resources in a wide variety of ways. A few of these are: to specify their menus and menu items, to define the appearance of windows and dialog boxes displayed by the applications, to hold strings and other localizable data, and to store fonts, sounds, pictures, custom cursors, and icons. These resources — let's call them *Macintosh resources* — are stored in the application's resource fork, which is automatically opened when the application is launched. The application can retrieve a particular resource by calling Resource Manager functions like `GetResource` and `GetIcon`.

Windows applications also use things called resources for many of the same purposes, but these resources are not the same as Macintosh resources. These resources — let's call them *Windows resources* — are stored inside the executable file (the .exe file) and can be loaded programmatically using functions like `LoadResource` and `LoadIcon`. Typically, an application's Windows resources are defined using a *resource script* (a file whose name ends in ".rc") and are automatically inserted into the executable file when the application is built.

As we've seen in many previous articles, it's often useful to use Macintosh resources in our QuickTime applications, whether those applications run on Macintosh or Windows computers. For instance, when we once wanted to elicit a URL from the user, we called `GetNewDialog` to display a dialog box defined by the application's resources; then we called `ModalDialog` to handle user actions in the dialog box. **Figure 1** shows the dialog box on Macintosh systems, and **Figure 2** shows the dialog box on Windows systems.

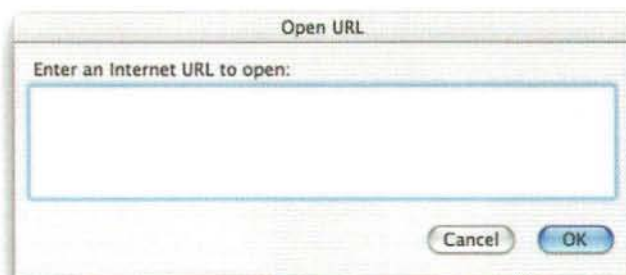


Figure 1: The Open URL dialog box (Macintosh)

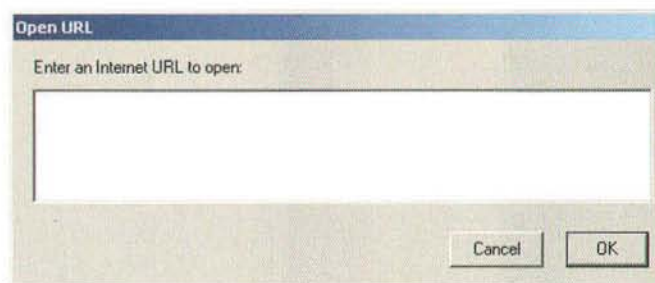


Figure 2: The Open URL dialog box (Windows)

Let's consider another example. Recently I added a properties panel to QuickTime Player, to display information about movie tracks. I constructed the panel on a Macintosh computer, using a text description that is converted by the tool Rez into a Macintosh resource. **Figure 3** shows the panel on the Mac OS 9, and **Figure 4** shows the panel on Windows. (This panel is not included with any shipping version of QuickTime Player, so don't bother looking for it.)

Tim Monroe is a member of the QuickTime engineering team. You can contact him at monroe@apple.com. The views expressed here are not necessarily shared by his employer.

MACS IN EDUCATION

THE CLASSROOM OF THE FUTURE

REVIEW > AN APPLE FOR THE TEACHER > IMAGE BY ANTHONY SAINT JAMES



CREATIVE DESIGNERS, WRITERS, MUSICIANS, BUSINESS LEADERS
AND OUR TECHNICAL EXPERT TEAM OFFER THEIR OWN PERSONAL
INTERPRETATION OF THINGS THAT ONLY THE MACINTOSH SYSTEM
CAN DELIVER. FEATURING OVER 240 PAGES OF REVIEWS, INTERVIEWS
PRODUCT NEWS, INSIGHTS, TRENDS AND THE LARGEST MACINTOSH
BUYERS' GUIDE. SUBSCRIBE > \$32/1 YEAR (4 ISSUES) OR \$62/2
YEARS (8 ISSUES): WWW.MACDIRECTORY.COM/PAGES/ADVERT.HTML
OR SEND CHECK OR MONEY ORDER TO: MACDIRECTORY SUB DEPT. 326
A STREET, 2C, SOUTH BOSTON MA 02110

MacDirectory

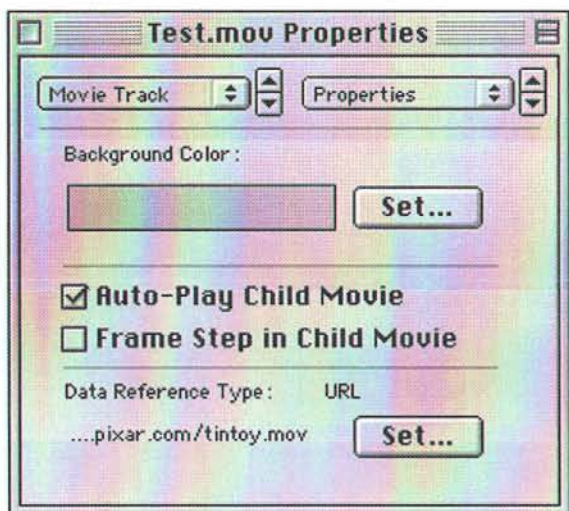


Figure 3: The Movie Track Properties panel (Mac OS 9)

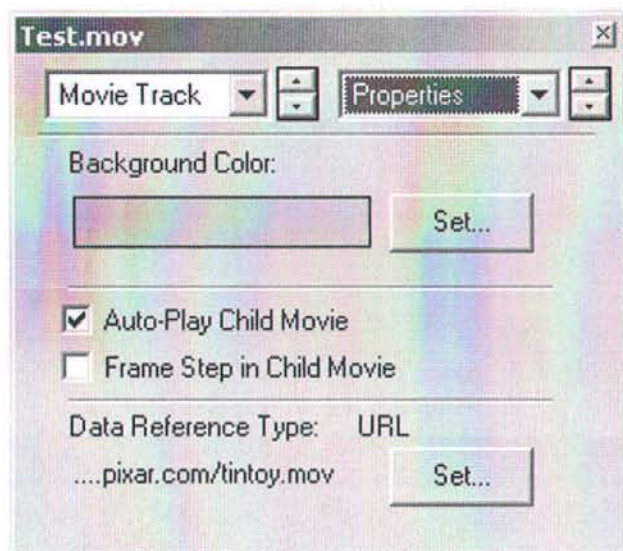


Figure 4: The Movie Track Properties panel (Windows)

The main advantage to using Macintosh resources for dialog boxes on both operating systems is that we can use the exact same resource description on both systems, and we can use the exact same code to display and manage the dialog boxes. On Windows, this magic is provided by the *QuickTime Media Layer* (or *QTML*), which we considered in an earlier article ("2001: A Space Odyssey" in *MacTech*, January 2001). In this article, I want to focus on how to attach Macintosh resources to a Windows application. We got a preliminary taste of doing this in that earlier article, where we saw how to use the tools Rez and RezWack on Windows to convert a resource description into a resource file and attach it to an application. Here, I want to investigate this process in a bit greater detail and to show how to configure

our Windows development environment, Microsoft Visual Studio, to perform these steps automatically each time we build our application.

Some programmers, of course, prefer to do most of their software development on Macintosh computers, so it's useful to see how to perform these steps on a Mac. Metrowerks' CodeWarrior integrated development environment (IDE) for Macintosh supplies compilers and linkers for Windows targets; moreover, the QuickTime software development kit (SDK) provides CodeWarrior projects configured to build Windows applications for virtually all of the sample code applications. All that's missing is a Macintosh version of the RezWack tool. In this article, we'll see how to fill in that gap. First we'll develop a standalone application that does this, and then we'll see how to construct a plug-in for the CodeWarrior IDE that performs the RezWack step as part of the build process. By the end of this article, you'll know how to create complete Windows applications that contain Macintosh resources, whether you prefer to program on Macintosh or Windows systems.

Before we begin, I should mention that having a single resource description for *all* of our target platforms is a wonderful goal that is not always achievable in practice, at least if we want to pay attention to Apple's human interface guidelines. The reason for this is simply that the Aqua appearance on Mac OS X has different layout requirements than the "classic" Mac OS 8 and 9 appearance. For instance, button labels are usually larger under Aqua than under earlier systems, so we need to make our buttons larger. **Figure 5** shows the movie track properties panel when displayed on a computer running Mac OS X. Here the entire dialog box is bigger, to be able to contain the larger controls and other dialog items.

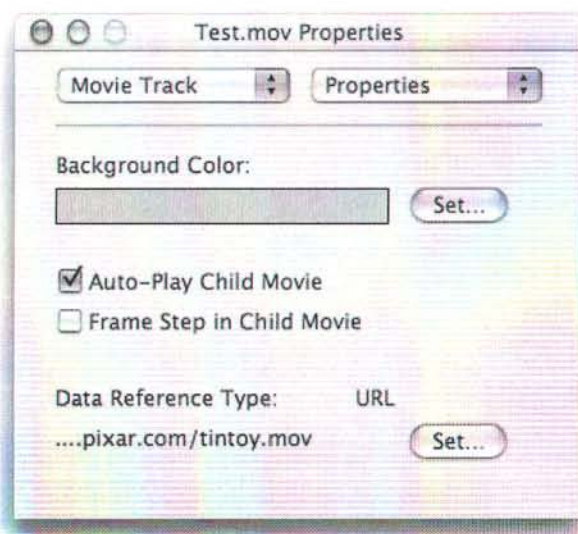


Figure 5: The Movie Track Properties panel (Mac OS X)

Creating Resource Files

Suppose, then, that we're writing a QuickTime application that is to run on Windows and we want to do our development on Windows using the Visual Studio environment. To make things as easy as possible, we'll construct our Macintosh resources by creating a text file that contains a resource description. For instance, **Listing 1** shows the resource description of the 'DITL' resource for the dialog box shown in **Figures 3 and 4**.

Listing 1: Resource description for the Movie Track Properties panel

```
resource 'DITL' (kResourceID) {
    /* array DITLarray: 12 elements */
    /* [1] */
    {8, 10, 25, 132},
    StaticText {
        disabled,
        "Background Color:"
    },
    /* [2] */
    {29, 12, 51, 138},
    UserItem {
        enabled
    },
    /* [3] */
    {30, 146, 50, 207},
    Button {
        enabled,
        "Set..."
    },
    /* [4] */
    {16, 7, 60, 214},
    UserItem {
        enabled
    },
    /* [5] */
    {74, 7, 90, 214},
    CheckBox {
        enabled,
        "Auto-Play Child Movie"
    },
    /* [6] */
    {94, 7, 110, 214},
    CheckBox {
        enabled,
        "Frame Step in Child Movie"
    },
    /* [7] */
    {120, 10, 136, 132},
    StaticText {
        disabled,
        "Data Reference Type: "
    },
    /* [8] */
    {120, 132, 136, 207},
    StaticText {
        disabled,
        "URL"
    },
    /* [9] */
    {140, 12, 156, 132},
    StaticText {
        enabled,
        ""
    },
    /* [10] */
    {138, 146, 158, 207},
    Button {
        enabled,
        "Set..."
    },
    /* [11] */
    {67, 10, 68, 215},
    UserItem {
        disabled
    },
    /* [12] */
    {115, 10, 116, 215},
    UserItem {
        disabled
    }
};
```

```
};
/* [6] */
{94, 7, 110, 214},
CheckBox {
    enabled,
    "Frame Step in Child Movie"
},
/* [7] */
{120, 10, 136, 132},
StaticText {
    disabled,
    "Data Reference Type: "
},
/* [8] */
{120, 132, 136, 207},
StaticText {
    disabled,
    "URL"
},
/* [9] */
{140, 12, 156, 132},
StaticText {
    enabled,
    ""
},
/* [10] */
{138, 146, 158, 207},
Button {
    enabled,
    "Set..."
},
/* [11] */
{67, 10, 68, 215},
UserItem {
    disabled
},
/* [12] */
{115, 10, 116, 215},
UserItem {
    disabled
},
};
```

The QuickTime SDK for Windows provides the tool Rez, which converts a resource description into a resource file. We can execute this line of code in a DOS console window to create a resource file:

Valentina

Object-Relational SQL Database

**The fastest database engine
for MacOS/Windows**

**It operates 100's and sometimes
a 1000 times faster than other systems**

www.paradigmasoft.com

Hosted by macserve.net
Download full featured evaluation version

Look Mom, No Wires!!!

Blue Berry Bluetooth USB Adapter



\$49⁹⁵

MacWireless 802.11b to USB

Connect any Mac with built-in USB ports to AirPort!

\$119⁹⁵



Macsense AERO PCMCIA Card

Get AirPort connectivity with your PCMCIA PowerBook!



\$89⁹⁵

Complete wireless solutions for Macintosh
Featured at DevDepot.com/wireless

Farallon/Proxim PCI SkyLINE Carrier Card

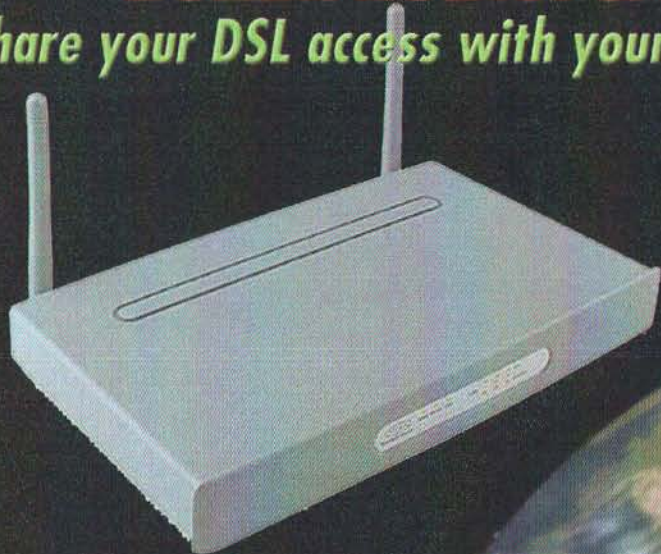
Use a SkyLINE AirPort Card in your PCI Desktop Mac!

\$39⁹⁵



Belkin Router & Wireless Gateway

Share your DSL access with your Ethernet and Wireless Computers



\$159⁹⁵

DEV DEPOT[®]

www.devdepot.com

PO Box 5200 •
Westlake Village, CA • 91359-5200
Voice: 877-DEPOT-NOW (877-337-6866)
Outside US/Canada: 805/494-9797 •
Fax: 805/494-9798
E-mail: orders@devdepot.com

Outside US/Canada: 805/494-9797 • Fax: 805/494-9798 • E-mail: orders@devdepot.com

```
QuickTimeSDK\QTDevWin\Tools\Rez
-i "QuickTimeSDK\QTDevWin\RIncludes" -i .
MIAMProperties.r -o MIAMProperties.qtr
```

Here, Rez converts the resource descriptions in the file `MIAMProperties.r` into resources in the resource file `MIAMProperties.qtr`. Rez looks in the current directory (.) and in the directory `QuickTimeSDK\QTDevWin\RIncludes` for any included .r files. (Of course, your paths may differ, depending on where you install the QuickTime SDK tools and .r files.)

The suffix ".qtr" is the preferred filename extension on Windows for files that contain Macintosh resources. When we are building an application, say `QTwiredActions.exe`, we should therefore name the resource file "QTwiredActions.qtr". In our application code, we need to explicitly open that resource file; to do this, we can use the code **Listing 2**.

Listing 2: Opening an application's resource file

```
WinMain
myLength = GetModuleFileName(NULL, myFileName, MAX_PATH);
if (myLength != 0) {
    NativePathNameToFSSpec(myFileName, &gAppFSSpec,
                           kFullPathName);

    gAppResFile = FSpOpenResFile(&gAppFSSpec, fsRdWrPerm);
    if (gAppResFile != kInvalidFileRefNum)
        UseResFile(gAppResFile);
}
```

Note that we do not need this code in our Macintosh applications; as we noted earlier, on the Mac an application's resource fork is automatically opened when the application is launched.

Embedding Resource Files in an Application

When developing and testing an application, it's okay to have to work with two files (the application file and the Macintosh resource file). But when we want to distribute an application, it's desirable to combine these two files into a single executable file. This is the job that RezWack is designed to accomplish. RezWack creates a new file that appends the resource data to the data in the executable file (and also appends some additional data to alert QTML to the fact that the .exe file contains some Macintosh resource data). We can call RezWack like this:

```
QuickTimeSDK\QTDevWin\Tools\RezWack -d QTwiredActions.exe
-r QTwiredActions.qtr -o TempName.exe
del QTwiredActions.exe
ren TempName.exe QTwiredActions.exe
```

RezWack does not allow us to overwrite either of the two input files, so we need to save the executable file under a temporary name, delete the original .exe file, and then rename the output file to the desired name. Once we've executed these commands, the file `QTwiredActions.exe` contains the original executable code

and the Macintosh resources that were contained in the file `QTwiredActions.qtr`.

Even if we have inserted the resource data into the executable file using RezWack, we still need to explicitly open the resource file at runtime; so our applications should always include the code in **Listing 2**, whether the resource data is in a separate file or is included in the executable file.

Adding a Post-Link Step

It would get tedious really fast to have to type the Rez and RezWack commands shown above in a DOS console window every time we rebuild our application. We can simplify our work by creating a batch file that contains these commands and by executing the batch file automatically as a custom post-link step. **Listing 3** shows the batch file that we use when building the debug version of the application `QTwiredActions.exe`.

Listing 3: Rezzing and RezWacking an application's resources

```
MyRezWackDebug.bat
REM *** batch program to embed our Macintosh
REM *** resources into our application file
..\..\QTDevWin\Tools\Rez -i "...\QTDevWin\RIncludes" -i .
QTwiredActions.r -o QTwiredActions.qtr
..\..\QTDevWin\Tools\RezWack -d .\Debug\QTwiredActions.exe
-r QTwiredActions.qtr -o .\Debug\TEMP.exe -f
del .\Debug\QTwiredActions.exe
REM *** now rename new file to previous name
ren .\Debug\TEMP.exe QTwiredActions.exe
```

You'll notice that the paths to the Rez and RezWack tools are relative to the location of the batch file, which we keep in the same folder as the Visual Studio project file; you may need to edit this file to set the correct paths for your particular folder arrangement.

We can tell Visual Studio to execute this file as a custom post-link step by adjusting the project settings. **Figure 6** shows the appropriate settings panel.

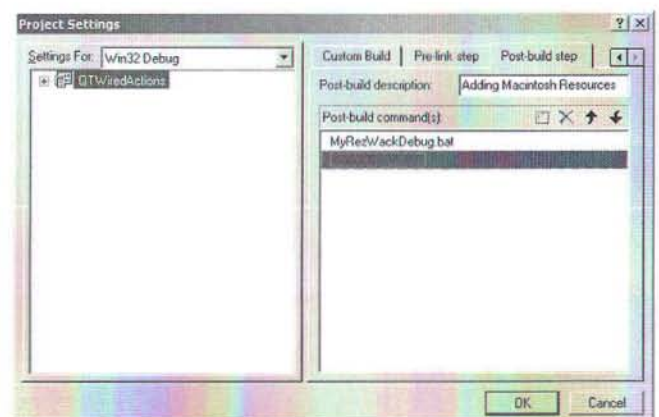


Figure 6: Setting a custom post-link step

DEVELOPMENT ON MACINTOSH

So what does RezWack actually *do*? We already know that it adds Macintosh resource data to a Windows executable file. But we need to uncover a bit more detail here if we are to be able to write a tool that performs resource wacking on Macintosh computers. Here's the essential information: RezWack creates a new file that begins with the data in the Windows executable file, followed immediately by the Macintosh resource data, padded to the nearest 4K boundary; this is all followed by some *RezWack tag data*. **Figure 7** illustrates the structure of a file created by RezWack; let's call this a *wacked file*.

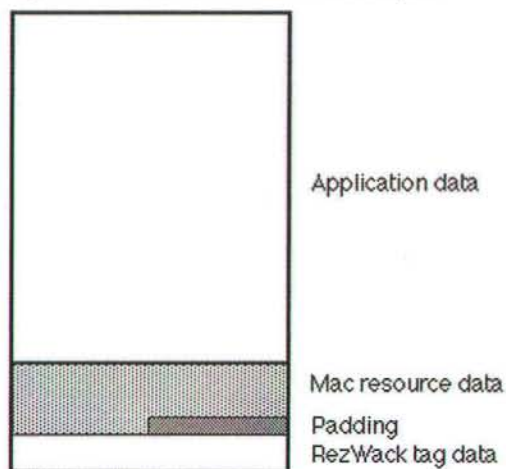


Figure 7: The structure of a wacked file

The RezWack tag data is a 36-byte block of data that describes the layout of the wacked file. It specifies the offset of the executable data from the beginning of the file (which is usually 0) and the size of the executable data; it also specifies the offset of the resource data from the beginning of the file and the size of the resource data. The tag data ends with this 12-byte identifier: "mkQuickTime™". (What's the "mk"? I strongly suspect that they are the initials of the engineer who implemented the RezWack tool.)

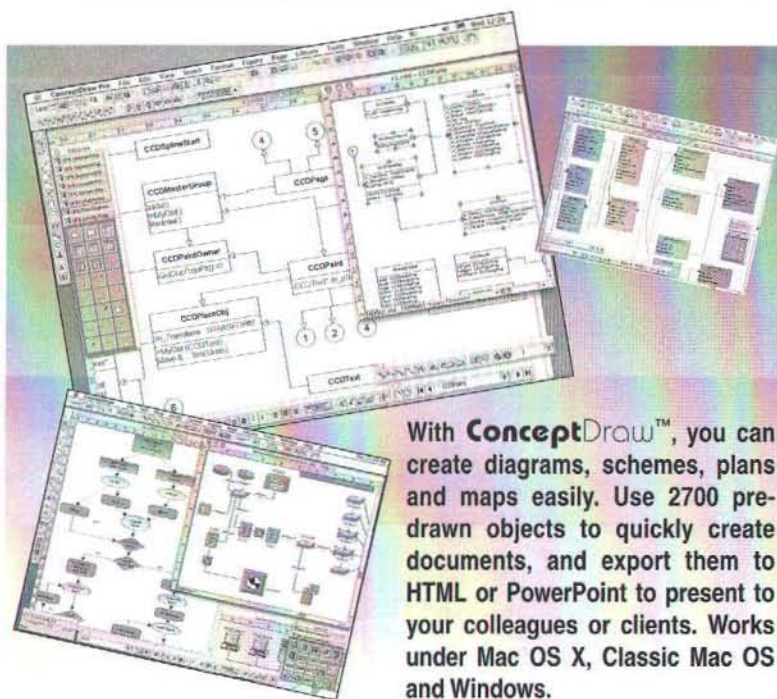
The idea here is that QuickTime can look at the last 12 bytes of a Windows executable file to determine whether it contains any Macintosh resources. If it does, then QuickTime can look at the last 36 bytes of the file to get the offset into the file of those resources and the size of the resource data.

Let's define a couple of constants that we can use to help us in the wacking process:

```
#define kRezWackTag          "mkQuickTime™"
#define kRezWackTagSize     12
```

Then we can define this data structure for the RezWack tag data:

```
typedef struct RezWackTagData {
    UInt32    fDataTag;           // must be 'data'
    UInt32    fDataOffset;       // offset of binary data
    UInt32    fDataSize;         // size of binary data
    UInt32    fRsrcTag;          // must be 'rsrc'
    UInt32    fRsrcOffset;       // offset of resource data
    UInt32    fRsrcSize;         // size of resource data
}
```



With **ConceptDraw™**, you can create diagrams, schemes, plans and maps easily. Use 2700 pre-drawn objects to quickly create documents, and export them to HTML or PowerPoint to present to your colleagues or clients. Works under Mac OS X, Classic Mac OS and Windows.

Use ConceptDraw Pro for:

- UML
- Network Diagrams
- Flowcharts
- Web Site Schemes
- Interface Modelling
- Process Flow



Download demo or order online at:
www.conceptdraw.com/professional
e-mail: sales@conceptdraw.com

Available at **DEPOT**
877-DEPOT-1000 or 800-444-9797
fax 800-444-9798
<http://www.depot.com>
order@depot.com

```

char      fWackTag[kRezWackTagSize];
RezWackTagData, *RezWackTagDataPtr;

```

We'll use this later, when we finally get around to building wacked files.

Setting up a Droplet

Our current goal is to develop a standalone application — let's call it “RezWack PPC” — that we can use as a droplet. That is, we'll compile our Windows code using CodeWarrior on the Macintosh and then drag the executable file created by CodeWarrior onto our droplet, which finds the appropriate resource file and creates a new file containing the executable data, the resource data, and the RezWack tag data.

We want the final wacked file to have the standard “.exe” suffix, so we need to tell CodeWarrior to generate an intermediate executable file with some other suffix. (Remember, RezWack won't overwrite either of its input files and so neither should we.) Let's use the suffix “.bin”, as shown in **Figure 8**.



Figure 8: Setting the intermediate file name

Our droplet should accept these binary files, look for a resource file with the suffix “.qtr”, and then create the final wacked file with the “.exe” suffix. To make RezWack PPC act like a droplet, we need to modify the function `QTApp_HandleOpenDocumentAppleEvent`. **Listing 4** shows our new definition of this function.

Listing 4: Handling files dropped onto RezWack PPC

```

PASCAL_RTN OSErr QTApp_HandleOpenDocumentAppleEvent
(const AppleEvent *theMessage, AppleEvent *theReply,
long theRefcon)
{
#pragma unused(theReply, theRefcon)

long      myIndex;
long      myItemsInList;
AEKeyword myKeyWd;
AEDescList myDocList;
long      myActualSize;
DescType  myTypeCode;

```

```

FSSpec      myFSSpec;
OSErr      myIgnoreErr = noErr;
OSErr      myErr = noErr;

// get the direct parameter and put it into myDocList
myDocList.dataHandle = NULL;
myErr = AEGGetParamDesc(theMessage, keyDirectObject,
                        typeAEList, &myDocList);

// count the descriptor records in the list
if (myErr == noErr)
    myErr = AECCountItems(&myDocList, &myItemsInList);
else
    myItemsInList = 0;

// open each specified file
for (myIndex = 1; myIndex <= myItemsInList; myIndex++)
    if (myErr == noErr) {
        myErr = AEGGetNthPtr(&myDocList, myIndex, typeFSS,
                            &myKeyWd, &myTypeCode, (Ptr)&myFSSpec,
                            sizeof(myFSSpec), &myActualSize);
        if (myErr == noErr) {
            FInfo myFinderInfo;

            // verify that the file type is kWinBinaryFileType or kWinLibraryFileType
            myErr = FSpGetFInfo(&myFSSpec, &myFinderInfo);
            if (myErr == noErr)
                if ((myFinderInfo.fdType == kWinBinaryFileType)
                    || (myFinderInfo.fdType == kWinLibraryFileType))
                    QTRW_CreateRezWackedFileFromBinary(&myFSSpec);
        }
    }

if (myDocList.dataHandle)
    myIgnoreErr = AEDisposeDesc(&myDocList);

QTRWFrame_QuitFramework(); // act like a droplet and close automatically
return(myErr);
}

```

As you can see, we look for files of type `kWinBinaryFileType` and `kWinLibraryFileType`, which have these file types:

```

#define kWinBinaryFileType    FOUR_CHAR_CODE('DEXE')
#define kWinLibraryFileType   FOUR_CHAR_CODE('IDL')

```

When we find one of these types of files, we call `QTRW_CreateRezWackedFileFromBinary` to do the necessary resource wacking. Notice that we call `QTRWFrame_QuitFramework` to quit the application once we are done processing the files dropped on it.

Wacking the Resources

Listing 5 shows our definition of `QTRW_CreateRezWackedFileFromBinary`; this function simply configures two additional file system specification records (one for the Macintosh resource file and one for the final wacked file) and then calls another function, `QTRW_RezWackWinBinaryAndMacResFile`, to do the real work. As indicated above, we assume that the resource file has the same name as the binary file but with the “.qtr” suffix and that the final wacked file has the same name as the binary file but with the “.exe” suffix. (I'll leave it as an exercise for the reader to implement less restrictive naming for the three files we need to work with.)

Listing 5: Setting up names for the resource and wacked files

```

QTRW_CreateRezWackedFileFromBinary

```

The Key is in Your Hands!

Does your dongle do all this?

Software Goes Online

Electronic Software Distribution – safely protected by WIBU-KEY.

Web Remote Programming

Reprogram the WIBU-BOX hardware at the customer's site directly via the Internet.

Web Authentication

Secure authentication via a two-way-encryption.

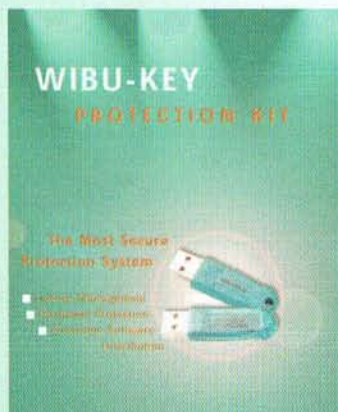
Pay-Per-Use

Usage dependent accounting.

Mac OS 9 & X

WIBU-KEY supports Mac OS, Windows and heterogeneous networks.

► **Yes? Then you are already using WIBU-KEY!**



► **No? Then order your Test Kit
at 1-800-986-6578**

**You will find more information at
www.griftech.com**

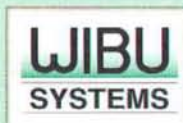


USA, Canada: Griffin Technologies, LLC
phone: (785) 832-2070 · fax: (785) 832-8787
email: sales@griftech.com · www.griftech.com

or visit our booth at...



...and see our products



WIBU-SYSTEMS AG
76137 Karlsruhe, Germany
WIBU-SYSTEMS USA, Inc.
Seattle, WA 98101
email: info@wibu.com

www.wibu.com

Test Kits also available at:

Argentina info@safeld-group.com, Australia simone.eckerle@wibu.com, Belgium wibu@impakt.be, Croatia aries@aries.hr, Denmark lean@danbit.dk, Egypt
esafwat@odec.com.eg, Finland finebyte@finebyte.com, France info@neol.fr, Hungary info@mrsoft.hu, Japan info@suncarla.co.jp, Jordan, Lebanon
starsoft@cyberia.net.lb, Korea dhkimm@wibu.co.kr, Luxembourg wibu@impakt.be, Netherlands wibu@impakt.be, Portugal dubit@dubit.pt, Syria
starsoft@cyberia.net.lb, Thailand preecha@dptf-th.com, United Kingdom info@codework.com, USA sales@griftech.com

```

OSErr QTRW_CreateRezWackedFileFromBinary
(FSSpecPtr theBinFSSpecPtr)
{
    FSSpec    myResSpec;
    FSSpec    myExeSpec;
    OSErr     myErr = paramErr;

    if (theBinFSSpecPtr == NULL)
        goto bail;

    // currently, we suppose that the Macintosh resource file has the same name as the
    // binary, but with "qtr" file name extension
    myResSpec = *theBinFSSpecPtr;
    myResSpec.name[myResSpec.name[0] - 2] = 'q';
    myResSpec.name[myResSpec.name[0] - 1] = 't';
    myResSpec.name[myResSpec.name[0] - 0] = 'r';

    // currently, we suppose that the Windows executable file has the same name as
    // the
    // binary, but with "exe" file name extension
    myExeSpec = *theBinFSSpecPtr;

    myExeSpec.name[myExeSpec.name[0] - 2] = 'e';
    myExeSpec.name[myExeSpec.name[0] - 1] = 'x';
    myExeSpec.name[myExeSpec.name[0] - 0] = 'e';

    myErr = QTRW_RezWackWinBinaryAndMacResFile(
        theBinFSSpecPtr, &myResSpec, &myExeSpec);

bail:
    return(myErr);
}

```

QTRW_RezWackWinBinaryAndMacResFile is the key function in RezWack PPC. It takes the binary file created by CodeWarrior and the Macintosh resource file and then creates the desired wacked file. First, it creates an empty file, having the same type and creator as the original binary file:

```

FSpGetFInfo(theBinFSSpecPtr, &myFileInfo);
FSpCreate(theExeFSSpecPtr, myFileInfo.fdCreator,
    myFileInfo.fdType, 0);

```

Then QTRW_RezWackWinBinaryAndMacResFile opens all three relevant files, using FSpOpenDF to open the binary and wacked files, and FSpOpenRF to open the resource file. Actually, we want our droplet to be a bit more flexible about handling resource files; it's possible for resource data to be stored in a data fork of a file, particularly if the resource data was created on a Windows computer (perhaps using the Rez tool). So first we'll attempt to open a resource fork having the appropriate name; if we can't find it or if its length is 0, we'll attempt to open a data fork having the appropriate name. **Listing 6** shows the code that accomplishes this.

Listing 6: Opening the resource file

```

QTRW_RezWackWinBinaryAndMacResFile
myErr = FSpOpenRF(theResFSSpecPtr, fsRdPerm, &myResFile);
if (myErr != noErr) {
    myTryDataFork = true;
} else {
    // it's possible that the resource fork exists but is 0-length; if so, try the data fork
    GetEOF(myResFile, &mySizeOfRes);
    if (mySizeOfRes == 0)
        myTryDataFork = true;
}

if (myTryDataFork) {
    // close the open resource file (presumably a 0-length resource fork)
    if (myResFile != -1)

```

```

        FSClose(myResFile);

    myErr = FSpOpenDF(theResFSSpecPtr, fsRdPerm, &myResFile);
    if (myErr != noErr)
        goto bail;
}

```

Once we've got all three files open, it's really quite simple to construct the wacked file. We copy the executable data from the binary file into the output file and copy the resource data from the resource file into the output file. Then we pad the file data to the nearest 4K boundary, as shown in **Listing 7**.

Listing 7: Padding the executable and resource data

```

QTRW_RezWackWinBinaryAndMacResFile
mySizeOfExe = mySizeOfBin + mySizeOfRes;
while ((mySizeOfExe + sizeof(myTagData)) % (4 * 1024) != 0)
{
    char    myChar = '\0';

    mySize = 1;
    myErr = FWrite(myExeFile, &mySize, &myChar);
    if (myErr != noErr)
        goto bail;
    mySizeOfExe++;
}

```

The last thing we need to do is append the RezWack tag data. **Listing 8** shows the code we use to do this. Notice that the tag data must be in big-endian byte order (as is typical for QuickTime-related data).

Listing 8: Appending the RezWack tag data

```

QTRW_RezWackWinBinaryAndMacResFile
myTagData.fDataTag = EndianU32_NtoB((long)'data');
myTagData.fDataOffset = 0L;
myTagData.fDataSize = EndianU32_NtoB(mySizeOfBin);
myTagData.fRsrcTag = EndianU32_NtoB((long)'rsrc');
myTagData.fRsrcOffset = EndianU32_NtoB(mySizeOfBin);
myTagData.fRsrcSize = EndianU32_NtoB(mySizeOfRes);
strncpy(myTagData.fWackTag, kRezWackTag, kRezWackTagSize);

mySize = sizeof(myTagData);
myErr = FWrite(myExeFile, &mySize, &myTagData);

```

And we're done! **Listing 9** shows the complete definition of QTRW_RezWackWinBinaryAndMacResFile.

Listing 9: Creating a wacked file

```

QTRW_RezWackWinBinaryAndMacResFile
OSErr QTRW_RezWackWinBinaryAndMacResFile
(FSSpecPtr theBinFSSpecPtr, FSSpecPtr theResFSSpecPtr,
    FSSpecPtr theExeFSSpecPtr)
{
    FInfo    myFileInfo;
    short    myBinFile = -1;
    short    myResFile = -1;
    short    myExeFile = -1;
    long     mySizeOfBin = 0L;
    long     mySizeOfRes = 0L;
    long     mySizeOfExe = 0L;
    long     mySize = 0L;
    long     myData = 0L;
    Handle    myHandle = NULL;
    RezWackTagData myTagData;
    Boolean    myTryDataFork = false;
    OSErr     myErr = paramErr;

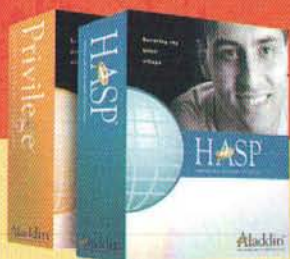
    // make sure we are passed three non-NULL FSSpecPtr's
}

```



CHICKENS SHOULD ROAM FREE. YOUR SOFTWARE SHOULDN'T.

Try our
Piracy Calculator.
See how much revenue may
have already flown the
coop at hasp.com/cal.



HASP & Privilege

Don't let "free roaming" software result in lost revenues — take advantage of the superior software licensing solutions offered by Aladdin Knowledge Systems. Did you know some businesses lose almost 50% of their revenue due to software piracy? That's why over 25,000 companies trust their software

licensing to Aladdin. Whether you prefer the proven reliability of HASP® dongles (USB or parallel port) or the state-of-the-art downloading of Privilege™, Aladdin products have a 99.97% success rate. Call us at 1-800-562-2543 to prevent your profits from flying the coop.

Aladdin®
SECURING THE GLOBAL VILLAGE
eAladdin.com

```

if ((theBinFSSpecPtr == NULL)
    || (theResFSSpecPtr == NULL)
    || (theExeFSSpecPtr == NULL))
    goto bail;

// get the creator and file type of the binary file
myErr = FSpGetFInfo(theBinFSSpecPtr, &myFileInfo);
if (myErr != noErr)
    goto bail;

// create the final executable file
myErr = FSpCreate(theExeFSSpecPtr, myFileInfo.fdCreator,
    myFileInfo.fdType, 0);
if ((myErr != noErr) && (myErr != dupFNErr))
    goto bail;

myErr = FSpOpenDF(theExeFSSpecPtr, fsRdWrPerm,
    &myExeFile);
if (myErr != noErr)
    goto bail;

// open the resource file; it's possible that the resources are stored in the data fork
// (particularly if the resource file was built on Windows); so make sure we've got a
// non-0-length resource file
myErr = FSpOpenRF(theResFSSpecPtr, fsRdPerm, &myResFile);
if (myErr != noErr) {
    myTryDataFork = true;
} else {
    // it's possible that the resource fork exists but is 0-length; if so, try the data fork
    GetEOF(myResFile, &mySizeOfRes);
    if (mySizeOfRes == 0)
        myTryDataFork = true;
}

if (myTryDataFork) {
    // close the open resource file (presumably a 0-length resource fork)
    if (myResFile != -1)
        FSClose(myResFile);

    myErr = FSpOpenDF(theResFSSpecPtr, fsRdPerm,
        &myResFile);
    if (myErr != noErr)
        goto bail;
}

myErr = FSpOpenDF(theBinFSSpecPtr, fsRdPerm, &myBinFile);
if (myErr != noErr)
    goto bail;

// copy the binary data into the final executable file
myErr = SetEOF(myExeFile, 0);
if (myErr != noErr)
    goto bail;

myErr = GetEOF(myBinFile, &mySizeOfBin);
if (myErr != noErr)
    goto bail;

myHandle = NewHandleClear(mySizeOfBin);
if (myHandle == NULL) {
    myErr = MemError();
    goto bail;
}

myErr = SetFPos(myBinFile, fsFromStart, 0);
if (myErr != noErr)
    goto bail;

myErr = FSRead(myBinFile, &mySizeOfBin, *myHandle);
if (myErr != noErr)
    goto bail;

myErr = SetFPos(myExeFile, fsFromStart, 0);
if (myErr != noErr)
    goto bail;

myErr = FWrite(myExeFile, &mySizeOfBin, *myHandle);
if (myErr != noErr)
    goto bail;

FSClose(myBinFile);

```

```

DisposeHandle(myHandle);

// copy the resource data into the final executable file
myErr = GetEOF(myResFile, &mySizeOfRes);
if (myErr != noErr)
    goto bail;

myHandle = NewHandleClear(mySizeOfRes);
if (myHandle == NULL) {
    myErr = MemError();
    goto bail;
}

myErr = SetFPos(myResFile, fsFromStart, 0);
if (myErr != noErr)
    goto bail;

myErr = FSRead(myResFile, &mySizeOfRes, *myHandle);
if (myErr != noErr)
    goto bail;

myErr = FWrite(myExeFile, &mySizeOfRes, *myHandle);
if (myErr != noErr)
    goto bail;

FSClose(myResFile);
DisposeHandle(myHandle);

// pad the final executable file so that it ends on a 4K boundary
mySizeOfExe = mySizeOfBin + mySizeOfRes;
while ((mySizeOfExe + sizeof(myTagData)) % (4 * 1024)
    != 0) {
    char myChar = '\0';

    mySize = 1;
    myErr = FWrite(myExeFile, &mySize, &myChar);
    if (myErr != noErr)
        goto bail;
    mySizeOfExe++;
}

// add on the special RezWack tag data
myTagData.fDataTag = EndianU32_NtoB((long)'data');
myTagData.fDataOffset = 0L;
myTagData.fDataSize = EndianU32_NtoB(mySizeOfBin);
myTagData.fRsrcTag = EndianU32_NtoB((long)'rsrc');
myTagData.fRsrcOffset = EndianU32_NtoB(mySizeOfBin);
myTagData.fRsrcSize = EndianU32_NtoB(mySizeOfRes);
strncpy(myTagData.fWackTag, kRezWackTag,
    kRezWackTagSize);

mySize = sizeof(myTagData);
myErr = FWrite(myExeFile, &mySize, &myTagData);

FSClose(myExeFile);

bail:
return(myErr);
}

```

CODEWARRIOR PLUG-INS

Our RezWack PPC droplet is a great little tool, but things would be even more convenient if we could get CodeWarrior to perform the resource wacking automatically — in the same way that we earlier added a post-link step to our Microsoft Visual Studio projects to run the Rez and RezWack tools automatically each time we build an application that uses Macintosh resources. Indeed this is possible, but it's significantly more complicated than just writing a batch file containing a few commands that are executed at the proper time. We need to write a *CodeWarrior plug-in*, a code module that extends the capabilities of the CodeWarrior IDE, to do this post-link step for us. We also will want to write a *CodeWarrior*

Multiple formats. Multiple platforms. Complex installers.

Aladdin solves the compression and installation puzzle.

**Trying to figure out how to handle multiple
compression formats and platforms?**

The StuffIt Engine solves the compression puzzle.



Aladdin's StuffIt Engine SDK:

- Adds value to your application by integrating powerful compression and encryption.
- Is the only tool that supports the StuffIt file format.
- Provides a single API that supports over 20 compression and encoding formats common on Macintosh, Windows, and Unix.
- Makes self-extracting archives for either Macintosh or Windows.
- Available for Macintosh, Windows, Linux, or Solaris.

Licenses start as low
as \$99/year

To learn more, visit:
www.stuffit.com/sdk/

StuffIt Engine SDK™ The power of StuffIt in your software.



**Looking for the easiest and fastest
way to build an installer?**

StuffIt InstallerMaker completes your puzzle.

It's not enough just to write solid code anymore. You still have to write an installer for your users. StuffIt InstallerMaker makes it simple and effective.

- StuffIt InstallerMaker gives you all the tools you need to install, uninstall, resource-compress or update your software in one complete, easy-to-use package.
- Add marketing muscle to your installers by customizing your electronic registration form to include surveys and special offers.
- Make demoware in minutes. Create Macintosh OS X and Macintosh Classic compatible installers with StuffIt InstallerMaker.

Prices start at \$250

To learn more, visit:
[www.stuffit.com/
installermaker/](http://www.stuffit.com/installermaker/)

StuffIt InstallerMaker™ The complete installation solution.™



www.stuffit.com
(831) 761-6200

© 2002 Aladdin Systems, Inc. StuffIt, StuffIt InstallerMaker, and StuffIt Engine SDK are trademarks of Aladdin Systems, Inc. The Aladdin logo is a registered trademark. All other products are trademarks or registered trademarks of their respective holders. All Rights Reserved.

settings panel plug-in, to allow us to specify the names of the resource file and the final output file. **Figure 9** shows our settings panel.



Figure 9: The RezWack settings panel

In this section, we'll take a look at the major steps involved in writing a CodeWarrior post-linker plug-in and a settings panel plug-in for wacking Macintosh resources into Windows applications. We won't have space to step through the entire process, but I'll try to touch on the most important points. Thankfully, most of the hard engineering is already done, for two reasons. First, we'll be able to use the `QTRW_RezWackWinBinaryAndMacResFile` function defined above (**Listing 9**) virtually unchanged within our post-linker plug-in code. And second, Metrowerks provides an SDK for writing CodeWarrior plug-ins that includes extensive documentation and several sample plug-in projects. Most of our work will consist of adapting two of the existing samples to create a RezWack post-linker and setting panel.

Writing a Post-Linker Plug-In

Let's begin by writing a post-linker plug-in. The first thing to do is download the CodeWarrior SDK from the Metrowerks web site. (See the end of this article for the exact location.) I installed the SDK directly into the "Metrowerks CodeWarrior" folder on my local machine. Then I duplicated the sample project called "Sample_Link". (There is no sample post-linker project, but it will be easy enough to convert their linker into a post-linker.) Let's call our new project "CWRezWack_Link". Once we rename all of the source code files and project files appropriately, we'll have the project shown in **Figure 10**. (The file `CWRezWack.rsrc` is new and contains a number of string resources that we'll use to report problems that occur during the post-linking; see the section "Handling Post-Link Errors" below.)

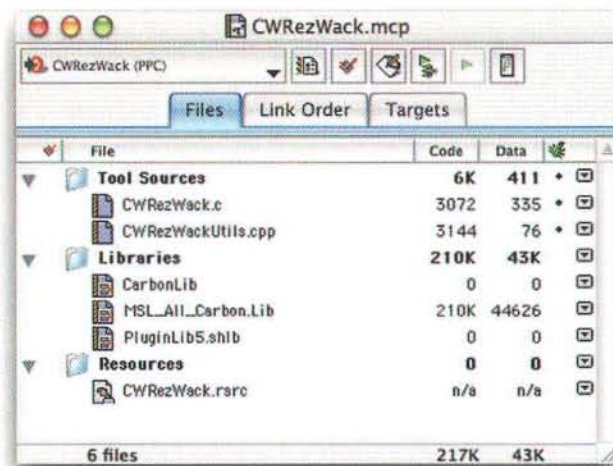


Figure 10: The RezWack post-linker project window

The sample linker project provided by Metrowerks includes both Macintosh and Windows targets, but I have removed the Windows target for simplicity; after all, we don't need a RezWack post-linker on Windows. As you can see, our project contains two source code files, `CWRezWack.c` and `CWRezWackUtils.cpp`. These are just renamed versions of the original `Sample_Link.c` and `SampleUtils.cpp`. We won't modify `CWRezWackUtils.cpp` further, except to update the included header file from `SampleUtils.h` to `CWRezWackUtils.h`.

In the file `CWRezWack.c`, we want to remove any code that supports disassembling. We'll remove the entire definition of the `Disassemble` function as well as its function declaration near the top of the file. Also, we'll rework the function `CWPlugin_GetDropInFlags` so that it looks like **Listing 10**.

Listing 10: Specifying the plug-in capabilities

```

CWPLUGIN_ENTRY(CWPlugin_GetDropInFlags)
    (const DropInFlags** flags, long* flagsSize)
{
    static const DropInFlags sFlags = {
        kCurrentDropInFlagsVersion,
        CWDROPINLINKERTYPE,
        DROPINCOMPILERLINKERAPIVERSION_7,
        isPostLinker | cantDisassemble, /* we are a post-linker */
        0,
        DROPINCOMPILERLINKERAPIVERSION
    };

    *flags = &sFlags;
    *flagsSize = sizeof(sFlags);

    return cwNoErr;
}

```

Originally, the fourth line of flags was just `linkMultiTargAware`; since we're not supporting Windows or disassembling, and since we are building a post-linker, we've changed that to `isPostLinker | cantDisassemble`.

Ever consider exhibiting at MacWorld?
Thought it might break your budget?

THINK AGAIN!

The most affordable solutions. Completely turnkey.

9 Pavilions to Choose From:

- Bluetooth & Wireless Technologies
- Enterprise, Networking & Server Solutions
- Education, Edutainment & Assistive Technologies
- MacTech Central
- Digital Media
- QuickTime
- International
- SciTech Pavilion
- Business Solutions

LIMITED TIME SPECIAL DISCOUNT OFFER:
Get \$1000 off your station package!

If you sign up by October 31, 2002, you'll get US\$1000 off your station, bringing the price to just US\$2995!
Send e-mail or visit the pavillion web site for more details.

reservations@xplain.com

<http://www.xplain.com/macworldexpo>

805/494-9797

Xplain Corporation • PO Box 5200 • 850-P Hampshire Road • Westlake Village, CA 91359-5200
Voice: 805-494-9797 • Fax: 805-494-9798 • reservations@xplain.com

The next important change concerns the routine `CWPlugin_GetTargetList`, shown in **Listing 11**. As you can see, we indicate that our post-linker applies only to applications built for the Windows operating system, since there is no need to RezWack Macintosh applications.

Listing 11: Specifying the plug-in targets

```

CWPlugin_GetTargetList
CWPLUGIN_ENTRY(CWPlugin_GetTargetList)
    (const CWTargetList** targetList)
{
    static CWDataType sCPU = targetCPUAny;
    static CWDataType sOS = targetOSWindows;
    static CWTargetList sTargetList =
        {kCurrentCWTargetListVersion, 1, &sCPU, 1, &sOS};

    *targetList = &sTargetList;

    return cwNoErr;
}

```

The main function of a CodeWarrior plug-in is largely a dispatcher that calls other functions in the plug-in in response to various requests it receives. Here we need to make just one change to the sample plug-in code, namely to return an error if our plug-in is asked to disassemble some object code:

```

case reqDisassemble:
    /* disassemble object code for a given project file */
    result = cwErrRequestFailed;
    break;

```

The only request we really want to handle is the `reqLink` request; in response to this request, we call the `Link` function defined in **Listing 12**.

Listing 12: Handling a link request

```

static CWResult Link(CWPluginContext context)
{
    CWTargetInfo    targetInfo;
    CWResult        err;
    FSSpec          fileSpec;

    // get the current linker target
    err = CWGetTargetInfo(context, &targetInfo);

    // get an FSSpec from the CWFileSpec
    ConvertCWFileSpecToFSSpec(&targetInfo.outfile,
        &fileSpec);

    // add Mac resources to linker target to create final Windows executable
    if (err == cwNoErr)
        err = CreateRezWackedFileFromBinary(context,
            &fileSpec);

    return (err);
}

```

`CWGetTargetInfo` returns information about the link target (that is, the file created by the CodeWarrior linker); we can extract a file system specification from that information using the utility function `ConvertCWFileSpecToFSSpec`. Then we pass that specification to `CreateRezWackedFileFromBinary`. Finally we are on familiar-looking ground once again. The

definition of `CreateRezWackedFileFromBinary` is shown in **Listing 13**.

Listing 13: Getting names for the resource and wacked files

```

CreateRezWackedFileFromBinary
OSErr CreateRezWackedFileFromBinary
    (CWPluginContext context, FSSpecPtr theBinFSSpecPtr)
{
    FSSpec        myResSpec;
    FSSpec        myExeSpec;
    CWMemHandle    prefsHand;
    SamplePref    prefsData;
    SamplePref    *prefsPtr;
    short         errMsgNum;
    CWResult       err;
    OSErr         myErr = paramErr;

    if (theBinFSSpecPtr == NULL)
        goto bail;

    // install a default name for the output file
    myExeSpec = *theBinFSSpecPtr;
    myExeSpec.name[myExeSpec.name[0] - 2] = 'e';
    myExeSpec.name[myExeSpec.name[0] - 1] = 'x';
    myExeSpec.name[myExeSpec.name[0] - 0] = 'e';

    // install a default name for the resource file
    myResSpec = *theBinFSSpecPtr;
    myResSpec.name[myResSpec.name[0] - 2] = 'q';
    myResSpec.name[myResSpec.name[0] - 1] = 't';
    myResSpec.name[myResSpec.name[0] - 0] = 'r';

    // load the panel prefs and get the specified names for the resource and output
    files
    err = CWGetNamedPreferences(context, kSamplePanelName,
        &prefsHand);
    if (err == cwNoErr) {
        err = CWLockMemHandle(context, prefsHand, false,
            (void**)&prefsPtr);
        if (err == cwNoErr) {
            prefsData = *prefsPtr;

            myExeSpec.name[0] = strlen(prefsData.outfile);
            BlockMoveData(prefsData.outfile, myExeSpec.name + 1,
                myExeSpec.name[0]);

            myResSpec.name[0] = strlen(prefsData.resfile);
            BlockMoveData(prefsData.resfile, myResSpec.name + 1,
                myResSpec.name[0]);

            CWUnlockMemHandle(context, prefsHand);
        }
    }

    myErr = RezWackWinBinaryAndMacResFile(theBinFSSpecPtr,
        &myResSpec, &myExeSpec, &errMsgNum);
    if (myErr != noErr)
        ReportError(context, errMsgNum);

bail:
    return(myErr);
}

```

The central portion of `CreateRezWackedFileFromBinary` retrieves the names of the resource file and the desired output file from our custom settings panel; then it calls `RezWackWinBinaryAndMacResFile` (which is largely just a renamed version of `QTRW_RezWackWinBinaryAndMacResFile`). In theory, we could omit the code that retrieves the resource file name and the output file name and just use hard-coded extensions, like we did in our RezWack PPC droplet. This



New OS. New Software. New Installer.

Introducing InstallAnywhere - Mac OS X Edition, the new power tool for your Mac OS X software installation. Stop using yesterday's tired old installer tools. Now, you can create industrial strength installers that are flexible, intuitive, and royalty-free. Your software will look better than ever and install perfectly, every time.

InstallAnywhere supports all Mac OS X features, like user authentication, file permissions, installing icons into the Dock, and offers a fully customizable Aqua look and feel.

Software innovators like Adobe, Apple, Borland, Gracion, LimeWire, and Sun already depend on Zero G for their software installation needs. See for yourself why InstallAnywhere - Mac OS X Edition is the new power tool for your software.

InstallAnywhere - The Industrial Strength Installer From Zero G.

Download a free trial version from <http://www.ZeroG.com/goto/mac>



would relieve us of having to write a settings panel plug-in, but it would provide less flexibility in naming things. Let's do things the right way, even if it means a bit of extra work.

Believe it or not, that's all we need to change in the sample linker to make our RezWack post-linker. We finish up by building the post-linker plug-in and installing it into the appropriate folder in the CodeWarrior plug-in directory. The next time we launch CodeWarrior, we'll be able to select our post-linker in the Target Settings panel of a Windows application project, as shown in **Figure 11**.

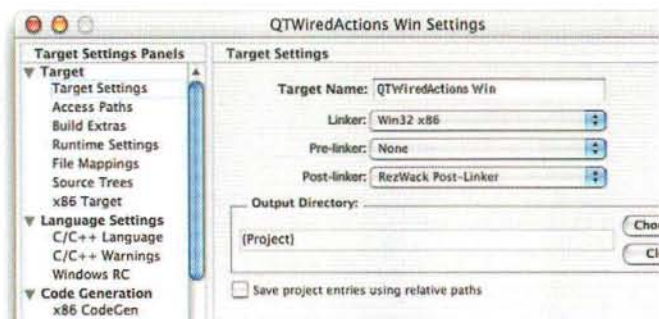


Figure 11: The post-linker menu with our RezWack plug-in

Handling Post-Link Errors

You may have noticed, in **Listing 13**, that `RezWackWinBinaryAndMacResFile` returns an error code through the `errMsgNum` parameter. For instance, if the attempt to open the specified resource file fails, then `RezWackWinBinaryAndMacResFile` executes this code:

```
if (myErr != noErr) {
    *errMsgNum = kOpenResError;
    goto bail;
}
```

If `CreateRezWackedFileFromBinary` sees that `errMsgNum` is non-zero, then it calls a function `ReportError` to display an error message to the user; a typical error message is shown in **Figure 12**.

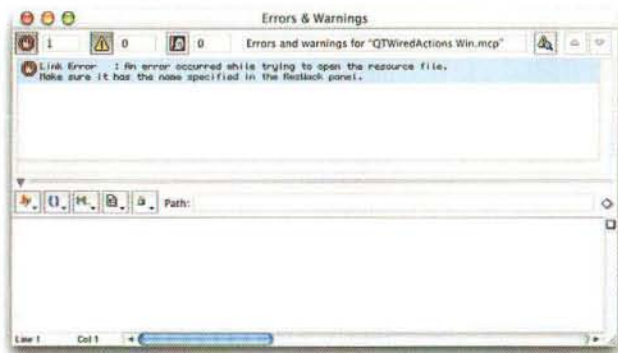


Figure 12: A post-linking error message

The `kOpenResError` constant is defined in our header file `CWRezWack.h`; here's the complete list of error-related constants defined there:

```
#define kExeCreateError      1
#define kExeCreateErrorL2   2
#define kOpenExeError       3
#define kOpenExeErrorL2     4
#define kOpenResError       5
#define kOpenResErrorL2     6
#define kOpenBinError       7
#define kOpenBinErrorL2     8
#define kGetMemError        9
#define kGetMemErrorL2     10
#define kWriteExeError      11
#define kWriteExeErrorL2    12
```

These are simply indices into a resource of type 'STR#' that is contained in the file `CWRezWack.rsrc`. Notice that each error has *two* corresponding string resources (for instance, `kOpenResError` and `kOpenResErrorL2`). These two strings provide the messages on the two lines of each error message in the "Errors & Warnings" window (for instance, "An error occurred while trying to open the resource file." and "Make sure it has the name specified in the RezWack panel.").

Listing 14 shows our definition of `ReportError`. The key ingredient here is the `CWReportMessage` function, which takes two strings and displays them to the user in the "Errors & Warnings" window.

Listing 14: Reporting a post-linking error to the user

```
void ReportError (CWPluginContext context, short errMsgNum)
{
    Str255    pErrorMsg;
    char      cErrorMsgL1[256];
    char      cErrorMsgL2[256];

    GetIndString(pErrorMsg, kErrorStrID, errMsgNum);
    if (pErrorMsg[0] != 0)
    {
        BlockMoveData(&pErrorMsg[1], &cErrorMsgL1, pErrorMsg[0]);
        cErrorMsgL1[pErrorMsg[0]] = 0;
    }

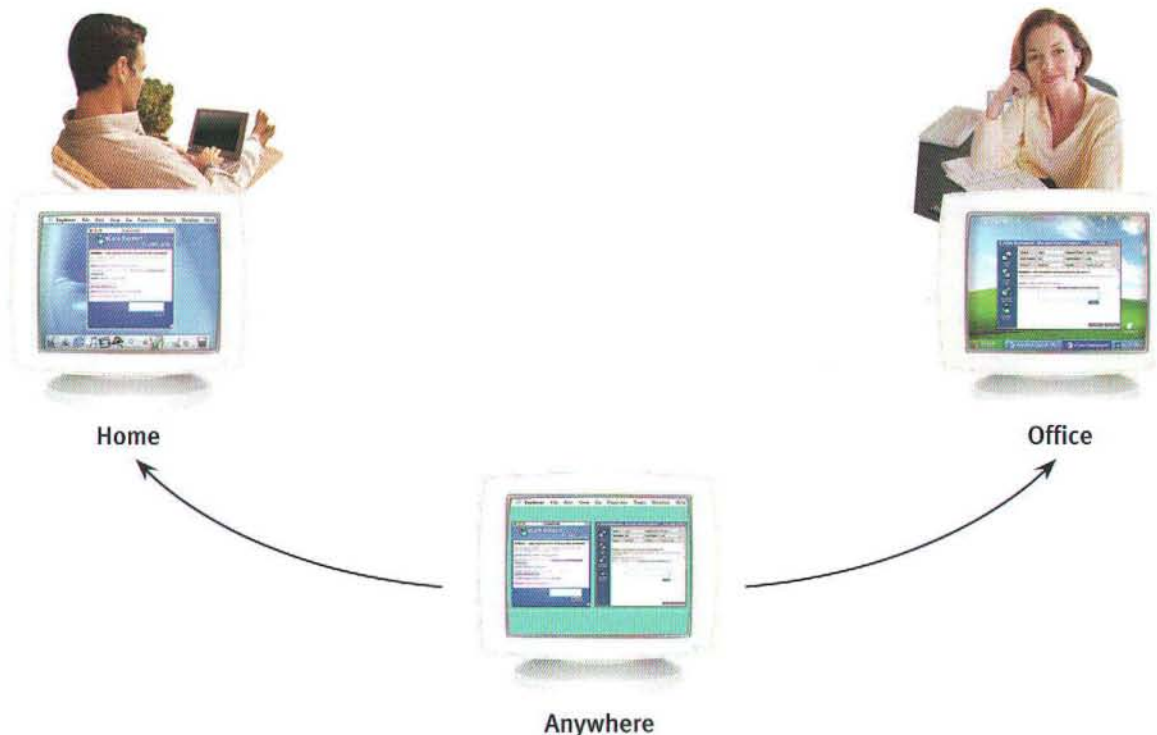
    GetIndString(pErrorMsg, kErrorStrID, errMsgNum + 1);
    if (pErrorMsg[0] != 0)
    {
        BlockMoveData(&pErrorMsg[1], &cErrorMsgL2, pErrorMsg[0]);
        cErrorMsgL2[pErrorMsg[0]] = 0;
    }

    CWReportMessage(context, NULL, (char*)&cErrorMsgL1,
                    (char*)&cErrorMsgL2, messageTypeError, 0);
}
```

Writing a Settings Panel Plug-In

The final thing we need to do is construct our settings panel plug-in (also called a *preference panel plug-in*), which displays the panel we saw earlier (**Figure 9**) and communicates the settings in that panel to our post-linker when it calls `CWGetNamedPreferences` (as in **Listing 13**). Once again, we'll clone the sample settings panel project and rename things appropriately, to obtain the project window shown in **Figure 13**.

Yes, you *can* be in two places at once.



For more than a decade, Timbuktu Pro and netOctopus have been the leading remote control, file transfer and systems administration applications for the Mac OS. Now, both of these indispensable tools are updated to take full advantage of the world's most advanced operating system.

Timbuktu Pro

Whether you're at home or at work, Timbuktu Pro allows you to operate distant computers as if you were sitting in front of them, transfer files or folders quickly and easily, and communicate by instant message, text chat, or voice intercom.

<http://www.timbuktopro.com>

netOctopus

Intuitive and powerful, netOctopus can manage a network of ten or 10,000 computers. Inventory computers, software and devices on your network; distribute software; configure remote computers; and create custom reports on the fly.

<http://www.netoctopus.com>

Learn more, try it, or buy it online. Call us at **1-800-485-5741**.



timbuktu pro® • netOctopus

netopia.

Long Distance

3.9¢ Per Minute!

Straight 6 second billing increments

Excellent rates on intrastate, intralata/toll calls and international calling with no term contract.

Toll Free (800/888/877/866) service, same low per minute rate for incoming calls.

10 cents per minute calling card.

Detailed billing directly from Capsule Communications, a Covista Company.

Quality electronic and telephone customer support.
No monthly billing fee if you sign up for AUTOPAY billing option or if your bill is over \$20.00 each month.

(NOTE: \$1.00 billing fee is charged when your bill is under \$20.00 for all non-Autopay customers.)



www.lowcostdialing.com

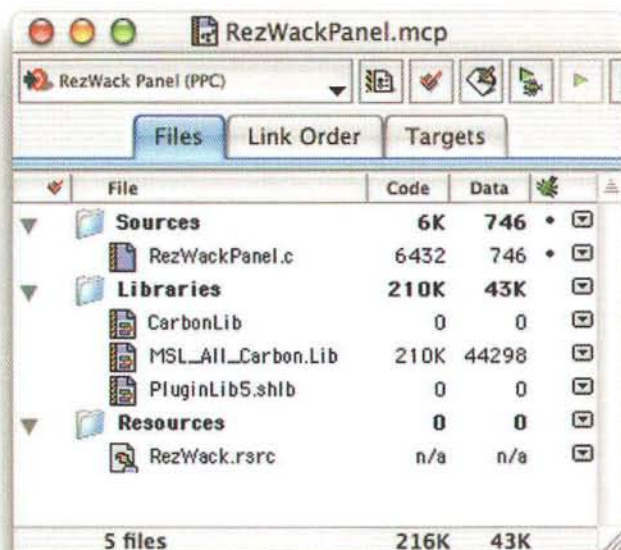


Figure 13: The RezWack settings panel project window

The layout of the settings panel is determined by a resource of type 'PPob' in the resource file RezWack.rsrc. Figure 14 shows the RezWack panel as it appears in the Constructor application.

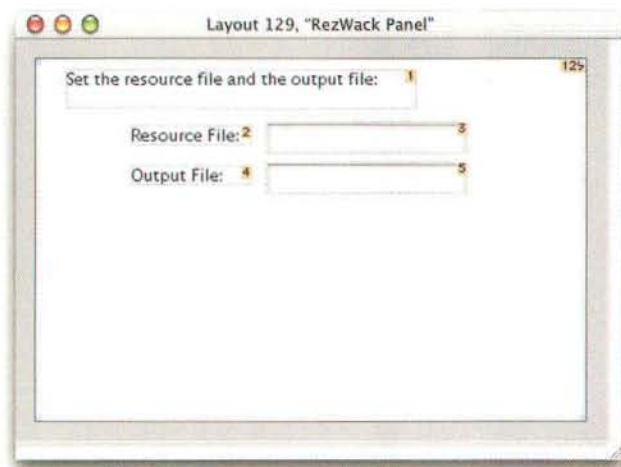


Figure 14: The RezWack settings panel in Constructor

Our RezWack settings panel contains only five items, which we can identify in our code using these constants:

```
enum {
    kStaticText          = 1,
    kResFileLabel,
    kResFileEditBox,
    kOutputFileLabel,
    kOutputFileEditBox
};
```

Most of the changes required in the sample settings panel plug-in code involve removing unneeded routines, which I

360 ONE VR



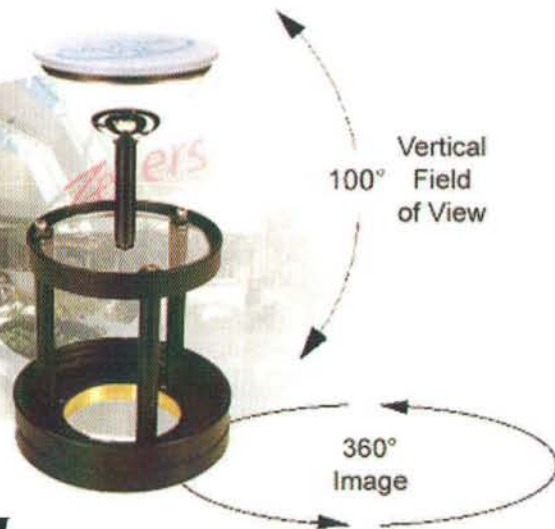
Macworld
**BEST
OF SHOW**
2002

The 360 One VR™ is an amazing product that captures a complete 360° panoramic image in a single camera shot. The system is the perfect solution for capturing all the action, as it happens! Years of extensive research have produced an innovative solution that creates an immersive image without the restrictions or compromises normally associated with panoramic photography.

The system consists of a lightweight and rugged proprietary optical device and the innovative PhotoWarp™ software from EyeSee 360. The unique mirrored optic provides a complete 360° horizontal coverage with an outstanding 100° vertical field-of-view (50° above and 50° below the horizon).

ONE SHOT

The optic has a standard 67mm thread mount and can be adapted to a number of cameras using either Kaidan camera specific mounting kits or commercially available step-up rings that fit your camera. For a complete list of supported cameras and kits, please visit the 360 One VR web site.



NO LIMITS!

The 360 One VR is available with a variety of photographic accessories. These include: monopods, tripods, bubble levels, camera bags and Pelican hard cases. You can purchase them separately or together at a discount in either our Pro Accessory or Real Estate Kit for the 360 One VR.

For more information on camera mounting kits and all the latest accessories, visit our website.



<http://www.360OneVR.com>

KAIDAN

Kaidan Incorporated

703 East Pennsylvania Blvd. Feasterville, PA 19053, USA

Phone: 215-364-1778 • Fax: 215-322-4186 • info@kaidan.com • <http://www.kaidan.com>



<http://www.eyesee360.com>

Kaidan is a trademark of Kaidan Incorporated. EyeSee360 and PhotoWarp are registered trademarks of EyeSee360 Inc. 360 One VR is a trademark of EyeSee360 and Kaidan Incorporated. QuickTime and Mac OS X are registered trademarks of Apple Computer, Inc. Windows is a registered trademark of Microsoft Corporation. Coolpix is a registered trademark of Nikon, Inc. Java is a trademark of Sun Microsystems. All other trademarks are the property of their respective owners. Specifications and equipment are subject to change without any notice or obligation on the part of the manufacturer. All panoramic images in this brochure were shot with the 360 One VR. © 2002 Kaidan Incorporated

won't discuss further. The two key functions are the `PutData` and `GetData` routines, which transfer data between the on-screen settings panel and an in-memory handle of settings data. For our `RezWack` settings panel, the data in memory has this structure:

```
typedef struct SamplePref {
    short    version;
    char     outfile[kFileNameSize];
    char     resfile[kFileNameSize];
} SamplePref, **SamplePrefHandle;
```

(Take a look back at **Listing 13** to see how we use the `outfile` and `resfile` fields when building a wacked file.)

Listing 15 shows our version of the `PutData` function, which copies settings information from the handle of settings data to the settings panel.

Listing 15: Copying settings data into the settings panel

```
static CWResult PutData (CWPluginContext context)
{
    CWResult    result = cwNoErr;
    CWMemHandle memHandle = NULL;
    SamplePref  thePreferences;

    try
    {
        // Get a pointer to the current prefs
        result = (GetThePreferences(context, &thePreferences));
        THROW_IF_ERR(result);
        // Stuff data into the preference dialog

        CWPanelsSetItemText (context, kOutputFileEditBox,
                             thePreferences.outfile);

        CWPanelsSetItemText (context, kResFileEditBox,
                             thePreferences.resfile);
    }
    catch (CWResult thisErr)
    {
        result = thisErr;
    }

    // Relinquish our pointer to the prefs data
    if (memHandle)
        CWWUnlockMemHandle(context, memHandle); // error is ignored

    return (result);
}
```

And **Listing 16** shows our version of the `GetData` function, which copies data from the panel to the handle of settings data.

Listing 16: Copying settings data out of the settings panel

```
static CWResult GetData(CWPluginContext context)
{
    CWResult    result = cwNoErr;
    CWMemHandle memHandle = NULL;
    SamplePref  thePreferences; // local copy of preference data
    char *      outname = (char *)malloc(255);
    char *      resname = (char *)malloc(255);
    short       i, j;

    try
    {
        // Get a pointer to the current prefs
        result = (GetThePreferences(context, &thePreferences));
        THROW_IF_ERR(result);

        // Stuff dialog values into the current prefs
        result = CWPanelsGetItemText(context,
```

```
        kOutputFileEditBox, outname,
        sizeof(thePreferences.outfile));
        THROW_IF_ERR(result);

        // apparently non-printing characters can squeeze their way into the dialog box,
        // so winnow them out...
        for (i = 0, j = 0; i < strlen(outname); i++)
            if (isprint(outname[i]))
                thePreferences.outfile[j++] = outname[i];

        thePreferences.outfile[j] = 0;

        result = CWPanelsGetItemText(context, kResFileEditBox,
        resname, sizeof(thePreferences.resfile));
        THROW_IF_ERR(result);

        // apparently non-printing characters can squeeze their way into the dialog box,
        // so winnow them out...
        for (i = 0, j = 0; i < strlen(resname); i++)
            if (isprint(resname[i]))
                thePreferences.resfile[j++] = resname[i];

        thePreferences.resfile[j] = 0;

        // Now update the "real" prefs data
        result = (PutThePreferences(context, &thePreferences));
        THROW_IF_ERR(result);
    }
    catch (CWResult thisErr)
    {
        result = thisErr;
    }

    // Relinquish our pointer to the prefs data
    free(outname);
    free(resname);

    return result;
}
```

I have found that non-printing characters can sometimes make their way into the edit-text boxes in the settings panel, so I explicitly look for them and remove them from the file names typed by the user.

The settings panel plug-in code contains a handful of other functions that save preferences on disk, restore preferences to their previous settings, and so forth. I'll leave the inspection of those routines to the interested reader. For the rest of us, we can finish up by building the settings panel plug-in and installing it into the correct folder in the CodeWarrior installation.

CONCLUSION

In this article, we've seen how to embed Macintosh resources into a Windows executable file, whether we want to develop our applications on Windows or Macintosh computers. This resource embedding allows us to take advantage of the support provided by the QuickTime Media Layer for those parts of the Macintosh User Interface Toolbox and the Macintosh Operating System that rely heavily on resources, including the Dialog Manager, the Window Manager, the Control Manager, and of course the Resource Manager. This in turn makes it easy to write our code once and deliver it on several platforms.

REFERENCES

You can download the CodeWarrior plug-in SDK from <http://www.metrowerks.com/MW/Support/API.htm>.

"Without a doubt, the Premiere Resource Editor for the Mac OS ... A wealth of time-saving tools."

— MacUser Magazine Eddy Awards

"A distinct improvement over Apple's ResEdit."

— MacTech Magazine

"Every Mac OS developer should own a copy of Resorcerer."

— Leonard Rosenthol, Aladdin Systems

"Without Resorcerer, our localization efforts would look like a Tower of Babel. Don't do product without it!"

— Greg Galanos, CEO and President, Metrowerks

"Resorcerer's data template system is amazing."

— Bill Goodman, author of *Smaller Installer* and *Compact Pro*

"Resorcerer Rocks! Buy it, you will NOT regret it."

— Joe Zobkiw, author of *A Fragment of Your Imagination*

"Resorcerer will pay for itself many times over in saved time and effort."

— MacUser review

"The template that disassembles PICT's is awesome!"

— Bill Steinberg, author of *Pyro!* and *PBTools*

"Resorcerer proved indispensable in its own creation!"

— Doug McKenna, author of *Resorcerer*



Resorcerer® 2

Version 2.0

The Resource Editor for the Mac™ OS Wizard

ORDERING INFO

Requires System 7.0 or greater,
1.5MB RAM, CD-ROM

Standard price: \$256 (decimal)

Website price: \$128 - \$256

(Educational, quantity, or
other discounts available)

Includes: Electronic documentation
60-day Money-Back Guarantee
Domestic standard shipping

Payment: Check, PO's, or Visa/MC
Taxes: Colorado customers only

Extras (call, fax, or email us):
COD, FedEx, UPS Blue/Red,
International Shipping

MATHEMAESTHETICS, INC.
PO Box 298
Boulder, CO 80306-0298 USA
Phone: (303) 440-0707
Fax: (303) 440-0504
resorcerer@mathemaesthetics.com

New
in
2.0:

- Very fast, HFS browser for viewing file tree of all volumes
- Extensibility for new Resorcerer Apprentices (CFM plug-ins)
- New AppleScript Dictionary ('aete') Apprentice Editor
- MacOS 8 Appearance Manager-savvy Control Editor
- PowerPlant text traits and menu command support
- Complete AIFF sound file disassembly template
- Big-, little-, and even mixed-endian template parsing
- Auto-backup during file saves; folder attribute editing
- Ships with PowerPC native, fat, and 68K versions

- Fully supported; it's easier, faster, and more productive than ResEdit
- Safer memory-based, not disk-file-based, design and operation
- All file information and common commands in one easy-to-use window
- Compares resource files, and even **edits your data forks** as well
- Visible, accumulating, editable scrap
- Searches and opens/marks/selects resources by text content
- Makes global resource ID or type changes easily and safely
- Builds resource files from simple Rez-like scripts
- Most editors DeRez directly to the clipboard
- All graphic editors support screen-copying or partial screen-copying
- Hot-linking Value Converter for editing 32 bits in a dozen formats
- Its own 32-bit List Mgr can open and edit very large data structures
- Templates can pre- and post-process any arbitrary data structure
- Includes nearly 200 templates for common system resources
- TMPLs for Installer, MacApp, QT, Balloons, AppleEvent, GX, etc.
- Full integrated support for editing color dialogs and menus
- Try out balloons, 'icb's, lists and popups, even create C source code
- Integrated single-window Hex/Code Editor, with patching, searching
- Editors for cursors, versions, pictures, bundles, and lots more
- Relied on by thousands of Macintosh developers around the world

To order by credit card, or to get the latest news, bug fixes, updates, and apprentices, visit our website...

www.mathemaesthetics.com

by Bob Boonstra, Westford, MA

AREA

As those of you who are regular readers know, the Programmer's Challenge problems have become more difficult over time. Just as all scientific discoveries worth making had been made by the mid-twentieth century, so it is that all simple Challenge problems have been posed and solved by this time. Then again, as they say, maybe not. This month's problem is borrowed from <http://www.polymathlove.com/>, where Gary Smith posts software he uses in teaching mathematics to elementary and middle school students. One of his programs is called Area Puzzles, where students create rectangles with specified areas to cover a grid subject to certain constraints.

The prototype for the code you should write is:

```
void Area(
    const short *cells,
    /* rectangle to be covered with smaller rectangles */
    /* index [row][col] as cells[row*rectWidth + col] */
    /* value N>0 means this cell must be covered by a rectangle of area N */
    short rectWidth,
    short rectHeight,
    Rect *yourRects[]
);
```

Your **Area** routine will be called with a rectangle of **cells** of width **rectWidth** and height **rectHeight**. Your task is to create a set of smaller rectangles (**yourRect**) that cover these cells. In doing so, you need to satisfy some constraints. Certain of the **cells** will have a nonzero value, and those **cells** must be covered by a rectangle with an area equal to that value. As an example, if the input **cells** were configured as follows ...

```
0 0 3 0 6 0 0 0 0 8
0 6 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 4 0 0
0 0 3 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
6 0 0 0 0 0 0 0 15 0
0 0 0 0 0 0 0 0 0 0
0 10 0 0 24 0 0 4 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 4 0 0 4 0 0 3
```

... you might create a set of rectangles like this, where each cell is shown with the number of the rectangle including that cell.

```
1 1 1 2 2 2 3 3 3 3
4 4 5 2 2 2 3 3 3 3
4 4 5 6 6 6 6 7 7 7
4 4 5 8 8 8 8 7 7 7
9 10 10 8 8 8 8 7 7 7
9 10 10 8 8 8 8 7 7 7
9 10 10 8 8 8 8 7 7 7
9 10 10 8 8 8 8 13 13 14
9 10 10 8 8 8 8 13 13 14
9 11 11 11 11 12 12 12 12 14
```

You should return the rectangles that cover the cell array and satisfy the constraints as **yourRects**. Each cell may be included in only one rectangle. If the cell has a nonzero value when **Area** is called, it

must be included in a rectangle with an area equal to that value. Memory for the rectangles you create will be allocated for you, and there will be as many of those rectangles as there are nonzero values in the **cells** array. Any solution that covers the entire **cells** array and satisfies the constraints will be considered correct.

Scoring will be based on execution time - the winner will be the solution that correctly solves the puzzles with the smallest execution time.

This will be a native PowerPC Carbon C++ Challenge, using the Metrowerks CodeWarrior Pro 7.0 development environment. Please be certain that your code is carbonized, as I may evaluate this Challenge using Mac OS X. Also, when submitting your solution, please include the project file and the code you used to test your solution. Occasionally I receive a solution that will not compile and, while I always try to correct these problems, it is easier to do so if I have your entire project available.

WINNER OF THE JULY, 2002 CHALLENGE

Congratulations to **Alan Hart** (United Kingdom) for winning the July One Time Pad Challenge. Recall that this Challenge required readers to decrypt a sequence of messages using a "one time pad". I place the term in quotation marks because the pad was neither "one time", as it was used multiple times, nor was it random, as a true one-time pad would be. Contestants had the advantage of possessing a dictionary of all the possible words in the communication.

Alan's solution tries each possible offset until the decoding attempt results in a sequence of words found in the dictionary. The speed of Alan's solution is due in part to his decision to test the decoding of the first four characters of the message for each offset against the dictionary before proceeding with the rest of the decoding. Another factor is Alan's reuse (with acknowledgement) of ideas from Ernst Munter's solution to the PlayFair Challenge, specifically the dictionary indexing approach. That approach creates an index for each word based on the first three characters that points to the first word in the dictionary beginning with those three letters. I'm pleased to see past Challenge code reused successfully.

Ernst Munter's second place entry also uses a brute force method. His approach is to select a possible offset from the pad, decrypt the message using that offset, verify that the decrypted message contains only words from the dictionary, and repeat with a new offset until successful. Ernst's solution also uses a modified version of the SpellTree dictionary class he developed for the PlayFair Challenge.

Jonny Taylor's third-place solution also examined the first three characters of the decoded message to determine whether an offset was promising enough to continue decoding. As noted by others, because the message may contain special characters not found in the dictionary, offsets rejected by this approach must be revisited to skip potential special characters if the message is not successfully decoded. Moses Hall takes a different approach, creating a finite state

Increase your projectivity.



Be productive. Stay productive.

Introducing FastTrack Schedule 8. Redesigned for Mac OS X and packed with new productivity features and a bold Aqua interface—FastTrack Schedule 8 has all the tools you need to ensure project success. For a free demo version or to order, call us today at 800.450.1983.



www.fasttrackschedule8.com



AEC
SOFTWARE

FastTrack Schedule 8 is a registered trademark of AEC Software. All other trademarks are the property of their respective owners. © 2004 AEC Software. Please call 800.450.1983 for information.

Macworld

Conference & Expo™



January 6–10

Conferences January 6 – 10, 2003

Expo January 7 – 10, 2003

San Francisco The Moscone Center

www.macworldexpo.com

For more information, call toll free
1-800-645-EXPO

Register online with
Priority Code: **A-MTO**

Find out what all the buzz is about at

www.macworldexpo.com

Only at Macworld Conference & Expo can the Mac community enhance their knowledge, network with peers and personally interact and purchase the latest products and technologies. Don't miss your chance to be a part of the #1 event for the Mac universe.

Acquire what you need (knowledge, products, services or solutions) to stay competitive and on the cusp of technology.

Enjoy the one-stop-shop atmosphere only Macworld Conference & Expo can provide you.

Test drive brand new products and services — be one of the first to kick the tires of the latest innovations.

Apply knowledge learned from 5 intense educational days at Macworld Conference & Expo immediately.

Network, exchange ideas and build contacts with like-minded, or not so like-minded, users and industry gurus.

Feel what it is like to be part of a loyal, powerful and holistic community.

Discuss your issues, mention your concerns, or praise the manufacturers of your favorite products directly.



Flagship Sponsors

Macworld

Macworld.com



MacCentral

Platinum Sponsor

COREL

machine encoding the dictionary. Rounding out the remainder of the five top-scoring entries, Jan Schotsman used the AliveC programming model and reports achieving a 5% increase in speed over the non-vectorized version.

The table below lists, for each of the solutions submitted, the number of test cases processed correctly, the execution time in seconds, the bonus awarded for code clarity and commentary, and the total score for each solution. It also lists the programming language of each entry. As usual, the number in parentheses after the entrant's name is the total number of Challenge points earned in all Challenges prior to this one.

Name	Cases Correct	Time (secs)	Bonus	Score	Lang
Alan Hart (39)	20	0.73	25%	5469.59	C++
Ernst Munter (872)	20	1.30	25%	9721.75	C++
Jonny Taylor (83)	20	1.67	25%	12499.44	C++
Moses Hall	20	2.54	15%	21572.92	C
Jan Schotsman (16)	20	3.05	5%	28959.52	C++
Tom Saxton (230)	20	3.08	5%	29268.66	C++
Damien Bobillot	15	12.11	15%	102918.96	C

TOP CONTESTANTS ...

Listed here are the Top Contestants for the Programmer's Challenge, including everyone who has accumulated 20 or more points during the past two years. The numbers below include points

awarded over the 24 most recent contests, including points earned by this month's entrants.

Rank	Name	Points (24 mo)	Wins (24 mo)	Total Points
1.	Munter, Ernst	251	8	882
2.	Saxton, Tom	65	2	230
3.	Taylor, Jonathan	64	2	90
4.	Stenger, Allen	53	1	118
5.	Wihlborg, Claes	40	2	49
6.	Hart, Alan	34	1	59
7.	Rieken, Willeke	22	1	134
8.	Landsbert, Robin	22	1	22
9.	Gregg, Xan	20	1	140
10.	Mallett, Jeff	20	1	114
11.	Cooper, Tony	20	1	20
12.	Truskier, Peter	20	1	20

Here is Alan's winning One Time Pad solution.

OneTimePad.cp
Copyright © 2002
Alan Hart

Problem definition:

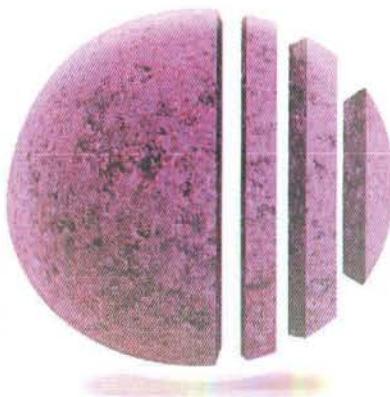
Decode multiple encrypted messages created using a known one time pad.

FREE UPGRADES FOR LIFE!

TIRED OF BEING SHAKEN DOWN FOR EXPENSIVE, BUGGY, FORCED UPGRADES FROM THE BIG VENDORS? JOIN THE PEOPLE WHO BELIEVE THAT YOU SHOULD BUY SOMETHING ONCE – AND HAVE IT FOREVER! SUPPORT AN OPEN AND FREE SOFTWARE DEVELOPMENT COMMUNITY NOT RUN BY MARKETING TYPES OR BEAN COUNTERS.



The applications in Stone Studio — starring Create® — have all the most used features of these applications: Illustrator, Quark Xpress, DreamWeaver, Freehand, Corel Draw, PageMaker, Timeslips, Acrobat Writer, Stuffit Deluxe, ImageReady, iPhoto and more. Visit www.stone.com/Stone_Studio_Successes to see how people just like you get their best ideas out into the world.



STONE STUDIO™

7 APPS 1 LOW PRICE 0 PAID UPGRADES

STONE STUDIO DOWNLOAD NOW FROM WWW.STONE.COM



\$299

Each message is encrypted using an unknown offset in the one time pad.
Each character in the encrypted message is the sum of the corresponding clear text and pad characters.
The sum is adjusted to remain in the valid character set range.

The character set is 62 upper/lower case alpha-numerics that appear in words in a case-insensitive dictionary, plus 33 other punctuation and special characters ("delimiters") that can appear in any locations between words.

Total time is to be minimized.

Assumptions:

We cannot make any assumptions about the number of delimiter characters that may precede the message or separate the words within it. In an unlikely extreme case the message could be a sea of delimiters with a few short words distributed anywhere within it. It is assumed that the test cases will not be pathological, and the majority of characters in the message will form dictionary words. In particular, the solution is optimized for messages with no leading delimiters before the first dictionary word. It decrypts messages with leading delimiters during a second scan of the pad.

Solution Summary:

The external interface calls are passed to a Decoder class which does the work.

The gDecoder instance is dynamically assigned by InitOneTimePad () and allocates a fixed size array of 256 KBytes for the main dictionary index. Further index Branch records are allocated dynamically during indexing. This adds a further 100KBytes to the space required in the case of the dictionary supplied with the test data.

The solution should fit comfortably in less than 500 KBytes heap space not including the dictionary and message strings.

The decoder creates a small static lookup table containing two concatenated copies of the character set to allow for wrap-around when subtracting the pad and cipher characters, allowing a simple lookup for decoding, and avoiding the need for range limiting.

Decoding is done by trying each possible pad offset in turn until the decode yields a sequence of words that are found in the dictionary.

During the first pass candidate pad offsets are found by testing the first 4 characters at each pad offset with the first 4 message characters. If this fails all pad offsets are retested on the whole message in case the first word does not start at the first character of the message. Decoding with each candidate pad is aborted if any invalid sequence of consecutive dictionary characters is detected or the end of the pad is reached.

The validity of a character sequence is tested in three stages:

1. A sequence of three or more characters must have a "header" index value matched by one or more dictionary words.
A shorter word must have an index value that matches a bit map of 1 and 2 character words.
2. Characters following a valid header index must form 4-character sequences that exist somewhere in the dictionary.
3. Finally the word is compared with the dictionary entries using a case insensitive character match and length comparison.

The dictionary index uses a 32 character (5 bit) enumeration to allow quick calculation of compact indices, and to enable bit mapping in a single 32 bit word. The 10 numeric characters are enumerated using 1 to 5, with pairs of digits sharing the same index value. 6 to 31 enumerate the alphabetic characters, ignoring case. Zero is used to denote any non-dictionary character. This enumeration means that words in the supplied dictionary containing numerals in the indexed characters may not be in correct index order. To avoid having to resort the dictionary this is handled in the index building and searching procedures.

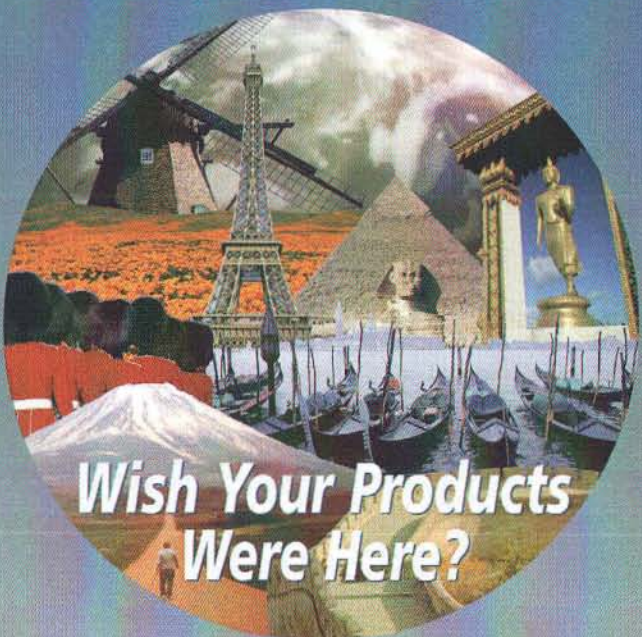
This mapping and indexing system allows a quick confidence check to be carried out during decoding so that non-indexed final dictionary searching is only done on longer words, and only when the word is likely to exist.

Words of one or two characters are recorded in a 32 x 32 bit map for a fast lookup. The index for a word of three or more characters is calculated by concatenating the index values of the first three characters.

The index selects one of an array of 32x32x32 (32K) Index records used for initial and final validation of decoded words.

If the Index value is the header for words longer than three characters it has a pointer to a dynamically allocated 32 entry lookup table. Each entry in the table points to the first word in the dictionary whose 4th character matches one character index value.

Classic Or Cocoa Applications? We Localize Them All!



- Translation • Engineering
- Localization • Project Management
- Desktop Publishing • Quality Assurance

To receive your FREE copy of
**The Guide to Translation and
Localization - Preparing Products
for the Global Marketplace:**

Call: **1-800-878-8523**

Fax: **1-503-419-4873**

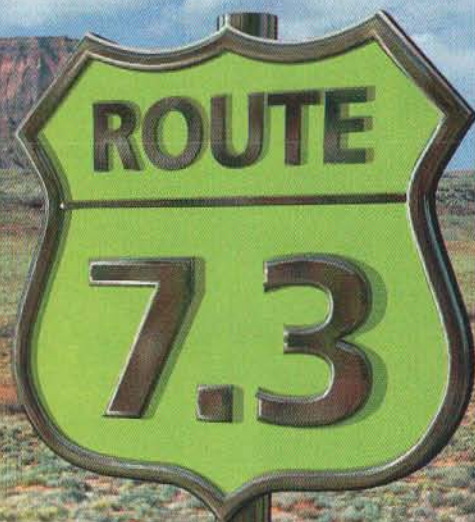
Email: **info@lingosys.com**

Or visit: **www.lingosys.com**



15115 SW Sequoia Pkwy. #200 Portland, OR 97224

ALL THE FREEDOM YOU NEED



SUSE LINUX 7.3

- SECURE
- STABLE
- SUPERIOR



SuSE Linux 7.3
PowerPC EDITION:
\$ 79,95

Yes, please send me the power-pack ...

... because Linux users enjoy the power and stability of SuSE Linux.
Don't compromise.

SuSE Linux 7.3 – no limits.

- Full integration with MOL (Mac-On-Linux)
- Scanning made easy with the KDE tool Kooka **NEW**
- Protect your data with Soft-RAID **NEW**
- Great sound with ALSA
- Surf safely with the Personal Firewall
- Professional installation support by phone, fax and e-mail
- etc.

Please bill the following address: ✂

Company _____

Last name/Name _____

Phone _____

Street _____

Zip-code/Postal-code _____

☐ SuSE Linux 7.3 PowerPC EDITION
\$ 79.95

SuSE Inc. — The Linux Experts
580 Second St.
Oakland · CA 94607

Order Online Today! <http://shop.suse.com/mt>



info@suse.com



(888) 875-4689



(510) 628-3381



The Index record also has a bit map of the valid fourth character indices that can follow this sequence when it appears at character positions 3, 6, 9 ... in the body of any word in the dictionary.

Optimizations applied included using unsigned types to remove compiled sign extend instructions, longs to remove compiled byte mask instructions, and the addition of the first pass decoder to limit the tested pad offsets to those yielding valid indices for the first 4 message characters. Instruction sequencing was also adjusted to minimize the time spent on processing failed pad offsets.

Acknowledgement

The dictionary indexing system uses some ideas gleaned by revisiting Ernst Munter's winning solution to the 1999 Playfair challenge.

```

//*****
#include "OneTimePad.h"
// use unsigned types wherever possible to minimize the insertion of sign extend
// instructions by the compiler
typedef unsigned char uchar;
typedef unsigned long ulong;
typedef unsigned long blong; // used instead of bool to avoid character masking
instructions

static const ulong IndexOffset [128] = {
// lookup table to convert character codes to index offset values
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
1, 1, 2, 2, 3, 3, 4, 4, 5, 5, 0, 0, 0, 0, 0, 0,
0, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 0, 0, 0, 0, 0,
0, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20,
21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 0, 0, 0, 0, 0
};

static inline ulong
IndexValue (const uchar* inWord) {
// Return the index for the first three characters of a word
ulong index = IndexOffset [*(inWord++)];
index = (index << 5) | IndexOffset [*(inWord++)];
return (index << 5) | IndexOffset [*(inWord)];
}

struct Branch {
const uchar** fFirstWord;
const uchar** fLastWord;
Branch () { fFirstWord = fLastWord = NULL; }

blong ValidateWord (uchar* inWord, ulong inWordLength) {
// Compare inWord with each word in this branch
// The indices are known to match, so inWord already points to the 4th character of
// the test word
// Note that the ambiguous indexing of numeric characters means it is possible to
// return a false positive.
// This is not considered a problem, as we are only trying to verify that the one time
// pad is correctly aligned to produce a string of credible words. However, it means
// that shorter dictionary words can appear within within the indexed range, and
// must be ignored.
const uchar** wordPtr = fFirstWord;
do {
ulong header = *(ulong*)(*wordPtr);
if ((header & 0x0ff0000) && (header & 0x0ff00)) {
// 3 or more characters in this dictionary word
const uchar* w1 = *wordPtr + 4;
uchar* w2 = inWord;
long difference = 0;
long len = inWordLength;
while (! difference && *w1) {
// compare the words until the end of the dictionary word
// converting upper case characters in inWord to lower case
difference = (*(w1++) - (*(w2++) | 0x20));
len--;
}
if (difference == 0 && len == 0) return true;
// words match
if (difference > 0) return false;
// this and subsequent words are greater than inWord
} while (wordPtr++ < fLastWord);
return false; // no match found
}
};

```

```

class Index {
// Dictionary index entry for a three-character sequence
public:
// Data encapsulation is not used, so that the decoder can access Index members
// directly for higher performance
Branch *fBranches;
// Dynamically allocated list of 32 pointers to words that start with this index
ulong fMap;
// Map of valid fourth character indices that can follow this index in the body
// of a word
// fMap bit 0 is used to register a valid three letter word for this index
Index () { fBranches = NULL; fMap = 0; }

~Index () { delete [] fBranches; }

ulong // Return the number of words registered for this index value
Register (
const uchar** inWordList, // Pointer to the first word to register
ulong inIndex, // The index value for words to register
const uchar** inLastWord, // The last word in the dictionary, for
// range checking

Index* inIndexArray
// The array if Index records, used to register word body sequences
) {
// This is the dictionary indexing procedure
// -Registers the existence of one or more 3 character words for this index in fMap
// bit zero
// -If there are words of 4 or more characters with this index value, allocates a
// Branch array and records the first and last words of the list for each 4th character.
// -Registers the existence of each 4-letter sequence in the body of each word in the
// fMap for its index
const uchar** wordPtr = inWordList;
ulong index = inIndex;
const uchar* word;
Branch* branch = NULL;
do {
uchar c = IndexOffset [*(wordPtr + 3)];
if (c) {
// 4 or more characters
if (fBranches == NULL) {
// allocate a new branch list to index words on the 4th character
fBranches = new Branch [32];
if (fBranches == 0)
return 0; // bail out and signal allocation failure
}
// record the start and end words in each branch
if (branch != fBranches + c) { // end of previous branch
if (branch)
// update the last word pointer for the old branch
branch->fLastWord = wordPtr - 1;
// move to the next branch
branch = fBranches + c;
if (branch->fFirstWord == NULL)
// this is the first word to be registered in this branch
// insert the first word pointer for this branch
branch->fFirstWord = wordPtr;
}
// register the remaining map bits for the body of this word
word = (*wordPtr) + 3; // skip the index
while (*word && *(word+1) && *(word+2)) {
// while there are three or more characters left
// get the 4th character index
c = IndexOffset [*(word + 3)];
if (c == 0) break; // no 4th character
// register the 4th character in the index bit map
inIndexArray [IndexValue (word) ].fMap |= (1 << c);
word += 3;
}
} else
// register the 3 character word in bit 0 of the index map
fMap |= 1;
wordPtr++; // next word

// until end of dictionary or index value changes
} while (wordPtr < inLastWord &&
index == IndexValue (*wordPtr));

if (branch) // update the last branch
branch->fLastWord = wordPtr - 1;
// return the number of words registered
return wordPtr - inWordList;
}

blong ValidateWord (uchar* inWord, ulong inWordLength)
{

```

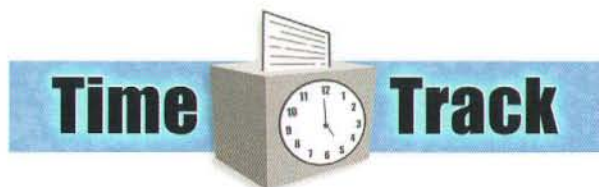


NameCleaner

An industrial-strength utility to manipulate file names and types, focused on preparing Macintosh files for use under Windows and Unix, and vice versa. Available with previewing, logging, scheduling and scripting - download a trial version from our website for OS 7/8/9 or OS X.

www.sigsoftware.com

Email: sales@sigsoftware.com



Keep track of every second of your time and bill for it with Time Track! Built in instructions make it easy to use. It is a simple way to manage your billable time for multiple projects and create a web page to show to your clients. Only \$24.95 per single user license per platform. Finally! A versatile time tracking solution for Macintosh, Windows, and Palm!

www.trinfinitysoftware.com



**The ultimate guide to the Classic Mac OS
& the Classic Environment in Mac OS X**

5 Mice/Stars—Macworld

Authoritative, a Must Have—MacAddict

Shareware of the Year—MacUser

The ultimate troubleshooting guide—Mac Today

The ultimate primer—ZDNet

www.InformINIT.com



Eudora Internet Mail Server (EIMS) 3.1 is the latest version of the most popular Internet mail server for the Macintosh. If you need to handle email for a dozen users, or thousands of users, EIMS is a reliable and easy to use solution.

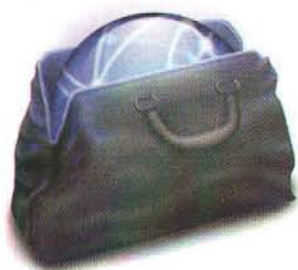
EIMS 3.1 is available for US\$400.00, there are no limits on the number of users that can be added, and free email support is included.

For more information, see
<http://www.eudora.co.nz/>



GraphicConverter converts pictures to different formats. Also it contains many useful features for picture manipulation.

See **www.lemkesoft.com**
for more information.



Watson 1.5

Winner of the 2002 Apple Design Award for Most Innovative Mac OS X Product! An Aqua experience for the most useful Web services: TV & Movie Listings, Reference, Translation, Stocks, Flights, Package Tracking, and more. With an open architecture for third-party tools as well. Download the demo now!

www.karelia.com/watson



Trapcode - plug-ins for Adobe® After Effects®

Trapcode Shine is a fast light effect plug-in. The effect looks very much like volumetric light, but is actually a 2D effect. There are special controls to make shimmering lights and numerous coloring modes. This is an effect that you see everyday on TV and in many movie titles. Shine is available for Mac, Mac OSX and Windows.

www.trapcode.com



By TLA Systems

The Dock with more than one dimension.

Create multiple docks of any size, assign hot keys and even put the Trash back on the Desktop. A flexible and feature-laden tool for power-users. Runs natively on Mac OS X and 9 in five languages.

"...you made the switch to OS X a lot easier for me..." - Bob LeVitus

"...DragThing can rightfully be called an indispensable aid to working with your Mac..." - MacUser UK

Download a copy now from www.dragthing.com.



Got a great product sold through Kagi?

Promote your product through the very cost effective Kagi Showcase in MacTech Magazine.

For more information, contact us at

adsales@mactech.com

```

// The dictionary lookup procedure.
// - Returns true if the word exists in the dictionary
if (inWordLength == 3)
// 3 character word. Check map bit 0
return (fMap & 0x01);
else if (fBranches) {
// Compare the 4th and subsequent characters with the words in the appropriate
// branch
Branch* branch = fBranches + IndexOffset [(inWord + 3)];
if (branch->fFirstWord)
return branch->ValidateWord (inWord + 4, inWordLength - 4);
}
return false;
}
};

```

```

class Decoder {

```

```

private:
// Pad information
const uchar* fPad; // local copy of the pad pointer
const uchar* fFirstPadChar;
// pointer to the current first pad character
const uchar* fLastPadChar;
// pointer to the last pad character to try
ulong fMessageLength; // length of the encrypted string
// Dictionary information
Index fIndexArray [ 32*32*32 ]; // the dictionary index array
ulong fShortWordMap [32];
// a bit map for one and two character words
// The decoder table
ulong fDecodeTable [190];
// lookup table to convert a pair of cipher/pad characters to a clear character
public
Decoder (const uchar* inPad)
{
// Constructor initializes the data members
// The dictionary index is built separately to allow Branch allocation failure to be
// handled gracefully
// initialize pad pointers and pad offset
fPad = inPad;
fFirstPadChar = fPad;

// Fill in the decoder table with two concatenated copies of the character set
int c;
ulong* t = fDecodeTable;
for (c = 0x20; c < 0x7f; c ++, t ++ )
*t = *(t + 0x5f) = c;

// Clear the short word map
for (c = 0; c < 32; c ++ )
fShortWordMap [c] = 0;
}

blong IndexDictionary (const uchar** inDictionary,
ulong inNumWords)
{
// Build the dictionary index and short word map
// Return false if Branch allocation fails
const uchar** wordPtr = inDictionary;
const uchar** lastWord = inDictionary + inNumWords;
ulong numWords;
do {
// Process the next index value
ulong index = *(ulong*)(*wordPtr);
numWords = 1;
if ((index & 0x0ff0000) == 0)
// Register a 1 character word in short word map zero
*fShortWordMap |= (1 <<
IndexOffset [(index >> 24) & 0xff]);
else if ((index & 0x0ff00) == 0)
// Register a 2 character word in the appropriate short word map
fShortWordMap [IndexOffset [(index >> 24) & 0xff]]
|= (1 << IndexOffset [(index >> 16) & 0xff]);
else {
// 3 or more characters
// Pass the word list to the appropriate Index record for registration
index = IndexValue (*wordPtr);
numWords = fIndexArray [index].Register (wordPtr, index,
lastWord, fIndexArray);
if (numWords == 0)
// branch list allocation failure - bail out
break;
}
} while (wordPtr++ != lastWord);
}
}

```

```

// Next word list
wordPtr += numWords;
} while (wordPtr < lastWord);

return (numWords > 0);
}

void FindPadOffset (const uchar* inEncryptedMessage,
uchar* outDecryptedMessage, ulong* outOffset)
{
// The main routine called to decode messages

// Use trial and error to find a pad offset that successfully decodes the message
// Return the offset found in *outOffset

// First Pass
// Optimized search for candidate pad offsets that decode a valid word index at the
// first encrypted character
// Build lookup tables to decode each of the first 4 message characters to its index
// value for any pad character
// Shift the pad through a 4-character buffer and apply this to the 4 message
// characters
typedef ulong PadMap [128];

ulong c0, c1, c2;
ulong* mapPtr;
const uchar* pad;
blong validOffset;
ulong map, i;
const uchar* cipher = inEncryptedMessage;
PadMap padMapArray [4];

// Measure the encrypted message length
while (*(++cipher)) {}
fMessageLength = cipher - inEncryptedMessage;

cipher = inEncryptedMessage;
for (map = 0; map < 4; map ++, cipher ++ ) {
// create the decode table for the next cipher character
mapPtr = padMapArray [map];
// clear the entries for invalid pad characters
*(mapPtr + 0x7f) = 0;
i = 0x20; while (i --) { *(mapPtr++) = 0; }
// set the decoded index values for each valid pad character
// calculate the first offset in the decode table for this cipher character
ulong* c = fDecodeTable + 0x3f + *cipher;
i = 0x5f; while (i --) { *(mapPtr++) = IndexOffset [(c - i)]; }
}

// Start at the beginning of the one time pad
// Decode these 4 characters with each pad offset and look for a valid word or index

fFirstPadChar = fPad;
validOffset = false;
pad = fPad;

// prime a sequence of 4 characters with the first 3 valid pad characters
ulong padBuffer = 0;
for (i = 0; i < 3; i ++ ) {
while (*pad < 0x20 || *pad > 0x7e) {
if (*pad == 0) return; // Abort if the pad is less than 4 characters
pad ++;
}
padBuffer = (padBuffer << 8) | (*pad ++);
}
do {
// Shift the next valid pad character into the set
while (*pad < 0x20 || *pad > 0x7e) { pad ++; }
padBuffer = (padBuffer << 8) | (*pad ++);

if ( (c0 = padMapArray[0] [padBuffer >> 24]) ) {
// 1st character decodes to a dictionary character
if ( (c1 = padMapArray[1] [(padBuffer >> 16) & 0xff]) ) {
// 2nd character decodes to a dictionary character
if ( (c2 = padMapArray[2] [(padBuffer >> 8) & 0xff]) ) {
// select the appropriate Index record for the 3 characters
Index* header = fIndexArray + (((c0 << 5) | c1) << 5) | c2;
// decode the 4th character
if ( (c0 = padMapArray[3] [padBuffer & 0xff]) )
// check that the Branch index exists for the 4th character
validOffset = header->fBranches &&
header->fBranches [c0].fFirstWord;
else
// check for a valid 3 character word
validOffset = header->fMap & 0x01;
} else

```

Borland®

#1 in Java™
Development
Solutions

Borland delivers leading Java application development technologies for building, optimizing, and deploying business-critical J2EE™ platform applications and Web Services.

Maximize the benefits of your enterprise Java solutions. From development to deployment, the award-winning Borland® software platform for Java enables you to increase productivity and performance while lowering the total

cost of ownership and reducing time-to-market. The result is strong, reliable applications that take full advantage of the power of Java.

Discover the key to success with Java technologies from the industry leader: Borland. Get your FREE *Enterprise Guide to Java Development* at info.borland.com/new/javasolutions/92125.html

Borland®

```

        // check for a valid 2 character word
        validOffset = fShortWordMap [c0] & (1 << c1);
    } else
    // check for a valid single character word
        validOffset = (*fShortWordMap) & (1 << c0);

    if (validOffset) {
        // This pad offset decodes a valid word or index
        // try to decode the whole message using this candidate pad
        validOffset = ApplyPad (inEncryptedMessage,
                                outDecryptedMessage);
    }
}
// move the pad offset to the next valid pad character
do {
    if (*(fFirstPadChar + fMessageLength))
        fFirstPadChar ++;
    else
        // We've run out of pad characters - end of pass 1
        goto SecondPass;
} while (*fFirstPadChar < 0x20 || *fFirstPadChar > 0x7e);

} while (! validOffset);

if (! validOffset) {
SecondPass:
    // Second Pass
    // The search for the first index may have failed due to leading delimiters
    // Try to decode the message using every pad offset in turn
    fFirstPadChar = fPad;
    do {
        validOffset = ApplyPad (inEncryptedMessage,
                                outDecryptedMessage);
        fFirstPadChar ++;
    } while (! validOffset && *(fFirstPadChar +
                                fMessageLength));
}
// return the offset
*outOffset = fFirstPadChar - fPad - 1;
}

ulong ApplyPad (const uchar* inEncryptedMessage,

```

```

                                uchar* outDecryptedMessage)
{
    // Called by FindPadOffset() with fFirstPadChar pointing to the start of a candidate
    // pad
    // Attempt to decode the message using this pad, and return success/failure

    const uchar* pad; // pointer to the current pad character
    const uchar* cipher; // pointer to the next encrypted character to decode
    uchar* clear; // pointer to the next clear character to decode
    ulong index; // index value of current word
    const ulong* origin = fDecodeTable + 0x5f;
    // pointer to the origin in the DecodeTable
    uchar* word; // pointer to start of current word being processed
    Index* body, *header; // pointers to the Index records for the
    // current word

    ulong c0, c1, c2;
    // index values of the first three characters of the current word
    ulong state = 0; // controls progress of the word decode process

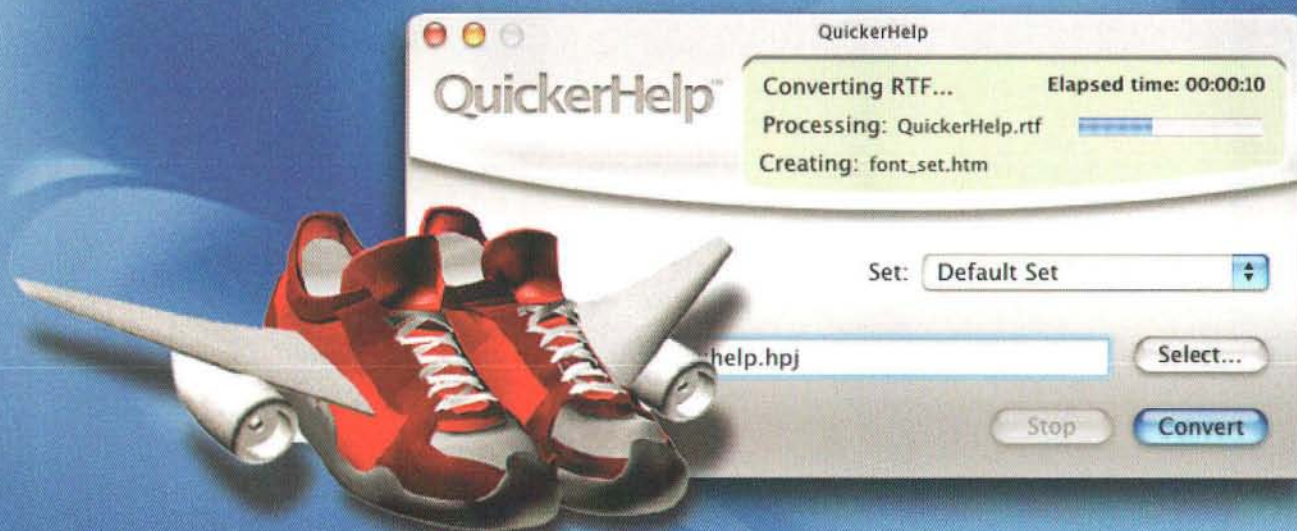
    clear = outDecryptedMessage;
    cipher = inEncryptedMessage;
    pad = fFirstPadChar;

    do {
        // for each character in the message

        // next valid pad character
        while (*pad < 0x20 || *pad > 0x7e) {
            if (*pad) pad++;
            else goto Exit; // end of pad
        }
        // decode the character
        *clear = *(origin + *cipher - *pad);
        switch (state) {
            case 0:// looking for 1st character of a word
                if ( (c0 = IndexOffset [*clear]) ) {
                    word = clear; // 1st character of a word
                    state = 1;
                }
                break;
            case 1:// looking for 2nd character of a word
                if ( (c1 = IndexOffset [*clear]) ) state = 2;
                // 2nd dictionary char

```

The fastest road to Apple Help starts here.



For moving your Help files to Apple Help, there's no faster, cleaner way than with QuickerHelp.
\$595 with discounts for ADC members. See www.mackiev.com/quickerhelp for details.



QuickerHelp™

The fastest road to Apple Help™


© 2002 The Software Mackiev Company. Software Mackiev, the Software Mackiev logo, QuickerHelp and the Flying Sneakers logo are trademarks of The Software Mackiev Company. Mac, the Mac logo and the QuickTime logo are trademarks of Apple Computer, Inc. registered in the U.S. and other countries. QuickTime, the QuickTime logo and the Built for Mac OS X graphic are trademarks of Apple Computer, Inc. used under license.

```

else if ((*fShortWordMap) & (1 << c0))
// valid 1 character word
state = 0;
else goto Exit;
break;
case 2: // looking for 3rd character of a word
if ( (c2 = IndexOffset [*clear]) ) {
header = fIndexArray + (((c0 << 5) | c1) << 5) | c2);
state = 3; // 3rd dictionary char
} else if (fShortWordMap [c0] & (1 << c1))
// valid 2 character word
state = 0;
else goto Exit;
break;
case 3: // looking for 4th character of a word
if ( (c0 = IndexOffset [*clear]) ) {
if (header->fBranches &&
header->fBranches [c0].fFirstWord) {
index = c0; // valid 4th dictionary char
body = NULL;
state = 4; // check body sequences
} else goto Exit;
} else if (header->fMap & 0x01)
// valid 3 character word
state = 0;
else goto Exit;
break;
case 4:
if ( (c0 = IndexOffset [*clear]) ) {
if (body == NULL) {
index = (index << 5) | c0; // accumulate the body index
if (index > 1024)
// third body index character. select the Index record
body = fIndexArray + index;
} else { // look up this character in the current body Index record
if (body->fMap & (1 << c0)) {
// valid sequence - start building the next body index
index = c0;
body = NULL;
} else goto Exit;
}
} else {
// end of the word, confirm it is in the dictionary
if (header->ValidateWord (word, clear - word))
state = 0;
else goto Exit;
}
default:
break;
}
pad ++;
clear ++;
} while ((*++cipher));

Exit:
*clear = 0; // terminate the decoded string
return ((*cipher) == 0); // successful if we reached the end of the
// message
}

} *gDecoder; // global pointer to a dynamically allocated instance

InitOneTimePad
void InitOneTimePad (const char *oneTimePad, const char
*dictionary[], long numDictionaryWords) {
// Create the decoder
gDecoder = new Decoder ((const uchar*) oneTimePad);
if (gDecoder) {
// Index the dictionary
if ( ! gDecoder->IndexDictionary ((const uchar**)
dictionary,
(ulong) numDictionaryWords)) {
// Allocation failure
delete gDecoder;
gDecoder = NULL;
}
}
}

DecryptedMessage
void DecryptMessage(const char *encryptedMessage, char
*decryptedMessage, long *offset) {
if (gDecoder)
gDecoder->FindPadOffset ((uchar*) encryptedMessage,
(uchar*) decryptedMessage, (ulong*)offset);
}

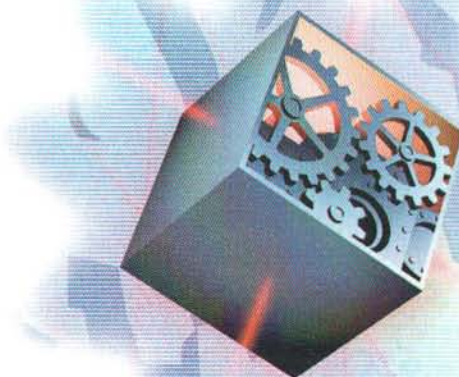
TermOneTimePad
void TermOneTimePad(void) {
delete gDecoder;
}

```

THE #1 RELATIONAL DATABASE ON MAC OS X

OPENBASE SQL

HIGH-PERFORMANCE RELATIONAL DATABASE



THE DATABASE THAT PAYS YOU CASH

*Announcing unlimited single-user
runtimes for just \$350 a year.*

Distribute your software with a royalty-free
single-user version of OpenBase SQL. Receive
monthly revenue when your users upgrade to one
of our low-cost multi-user subscription licenses.

Go to www.openbase.com/cashback for more details!

Visit our Web site and check out the innovative features that make
OpenBase SQL the database of choice for developers.

www.openbase.com



OPENBASE
INTERNATIONAL

CREATE WITH POWER™



By Dan Wood, Alameda CA

Table Techniques Taught Tastefully (part 2)

Using NSTableView for Real-World Applications

INTRODUCTION

Part one of this series of articles introduced the wonderful NSTableView class in Cocoa, going over the basics—displaying textual data, adding and deleting rows, and adjusting the columns of a table. With those techniques, you should be able to write code with some pretty useful displays. But there's so much more you can do with NSTableView, and this part of the series goes into some intermediate techniques that will help you feel like a "Table Jockey."

In this article, we'll cover a little bit more in the topic of deleting rows that we introduced in part 1. Then we'll introduce alphabetic type-ahead, which allows the user to start typing the first few letters of a row's relevant text to get the selection established. Next, we'll cover several aspects of sorting and reordering a table's rows, including sorting by clicking on a column header and drag-and-drop rearrangement. Finally, we'll cover exporting of data from a table via drag-and-drop and via the clipboard.

Be sure to follow along with the "TableTester" application (downloadable at www.karelia.com/tabletester/), a program showing off most of the table features described in this series. It contains the source code corresponding to the techniques in this article as well as those in part one, in case you missed it the first time around.

DELETING ROWS (THE SEQUEL)

In part 1, we discussed how to delete the selected rows in a table, responding to a button or the Clear menu. How about deleting the selected rows when the delete key is pressed? This requires us to create a new subclass of NSTableView and override the `keyDown:` method to pass along the same `deleteSelectedRowsInTableView:` method to the data source. To create the user interface for this, we create a table but then set

its custom class to our subclass, `DeletableTableView`, using the class inspector in Interface Builder.

Listing 1: DeletableTableView.m

keyDown:
Trap out delete keys and pass along the method to delete the selected rows. Otherwise, just let the superclass handle the event. The table must be "first responder" for this to be processed.

```
- (void)keyDown:(NSEvent *)theEvent
{
    NSString *keyString
        = [theEvent charactersIgnoringModifiers];
    unichar keyChar = [keyString characterAtIndex:0];

    switch (keyChar)
    {
        case 0177: // Delete Key
        case NSDeleteFunctionKey:
        case NSDeleteCharFunctionKey:
            if ([self selectedRow] >= 0 && [[self dataSource]
                respondsToSelector:
                    @selector(deleteSelectedRowsInTableView:)])
            {
                [[self dataSource]
                    deleteSelectedRowsInTableView:self];
            }
            break;
        default:
            [super keyDown:theEvent];
    }
}
```

A feature not implemented above is undoability; we'll leave this as an exercise for the reader.

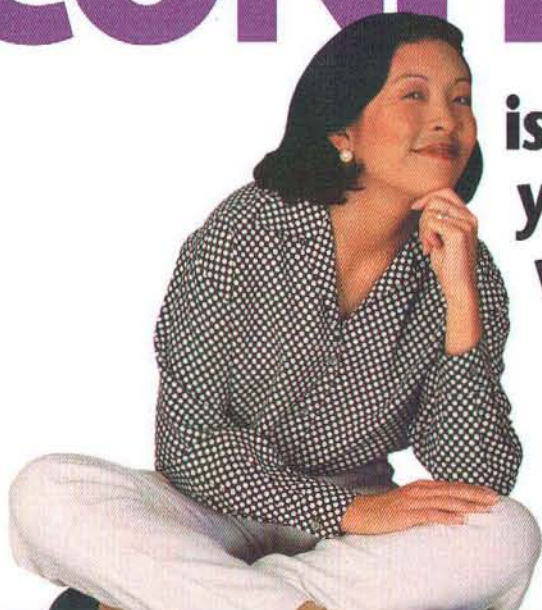
ALPHABETIC TYPE-AHEAD FOR TABLE SELECTION

When lists are long, keyboard navigation — the ability to start typing on the keyboard to navigate to the desired elements in a list — is quite convenient. This functionality isn't built into NSTableView, but it is possible by creating a subclass that pays attention to the keystrokes when the table is active.

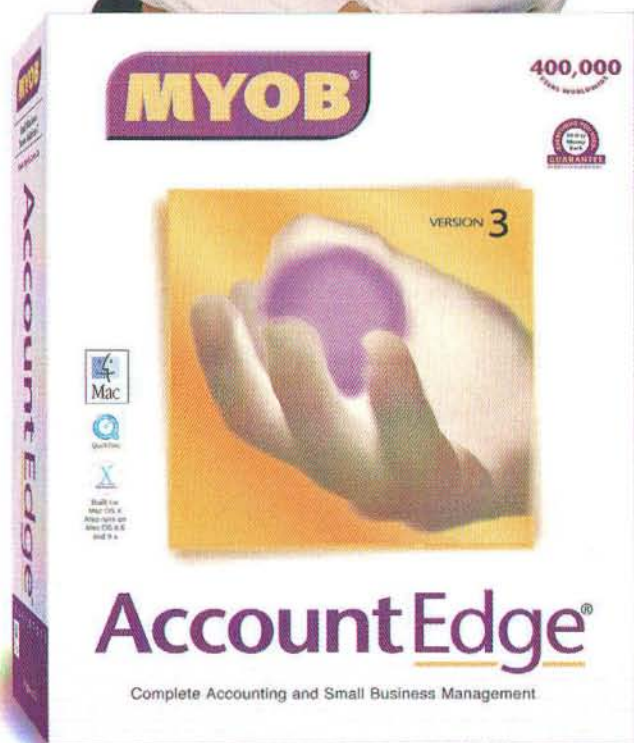
The algorithm is fairly straightforward. When the user presses a key, the key is appended to a string, and that string is used to try to select the closest row in the table. As more keys

Dan Wood once took an introductory Arabic class, but nobody in the room knew what language they were being taught. He likes to buy fruits and vegetables from the farmer's market on Tuesday mornings. He missed the last two days of WWDC this year due to the birth of his son. He is the author of *Watson*, an application written in Cocoa. Dan thanks Chuck Pisula at Apple for his technical help with this series, and acknowledges online code fragments from John C. Randolph, Stéphane Sudre, Ondra Cada, Vince DeMarco, Harry Emmanuel, and others. You can reach him at dwood@karelia.com.

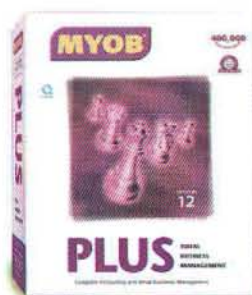
CONFIDENCE



is knowing
you've got
what it takes!



what it takes
to
manage your small business
successfully.



Award Winning Small Business Accounting and Management Software

Join the over 400,000 MYOB customers worldwide who trust their business success to MYOB products. From business basics to complete management, MYOB has what it takes to Mind Your Own Business better. Available for Macintosh and Windows.

Visit us online for a **FREE Trial Version**

www.myob.com/us/mt

800-322-MYOB(6962)



**Small Business.
Smart Solutions.**

are pressed, the string builds up, and the selection becomes more refined. If a time interval of a couple of seconds passes with no keystrokes, the typing buffer is cleared, so the user can make a fresh selection.

To handle the keystrokes, we override `keyDown:` to merely send the `interpretKeyEvents:` message to self. According to the `NSResponder` documentation, this is how to intercept keystrokes; it will cause the `insertText:` method to be invoked, which we handle below.

Listing 2: TypeaheadTableView.m

keyDown:
Indicate that the keystrokes from the event should be interpreted. The table needs to be "first responder" for this to be effective.

```
- (void)keyDown:(NSEvent*)inEvent
{
    [self interpretKeyEvents:
     [NSArray arrayWithObject:inEvent]];
}
```

We then implement `insertText:` to add the typed characters to our buffer of keystrokes, and go select the appropriate row based on what has been typed so far. Thinking in terms of Model-View-Controller partitioning, we send a message from the view (the `NSTableView` subclass) to the Controller (the table's delegate) to perform the selection, by invoking `typeAheadString:inTableView:` on the delegate. We will provide a sample implementation later.

We also queue up a message to send in the near future to clear out the keystroke buffer. But what is the magic number for the time delay before the buffer is cleared? You could hard-wire a constant, but that might not satisfy all users. Instead, you can base the time delay on the "Delay Until Repeat" setting in the System Preferences. The chapter from *Inside Macintosh* (Remember *Inside Macintosh*?) on the List Manager recommended two times that threshold value, but no more than two seconds. Whereas Carbon programs get the delay from the low memory global function `LMGetKeyThresh()`, this value is available via the defaults mechanism via the "InitialKeyRepeat" key.

Listing 3: TypeaheadTableView.m

insertText:
Process the text by adding it to the typeahead buffer and selecting the appropriate row.

```
- (void)insertText:(id)inString
{
    // Make sure delegate will handle type-ahead message
    if ([[self delegate] respondsToSelector:
        @selector(typeAheadString:inTableView:)])
    {
        // We clear it out after two times the key repeat rate "InitialKeyRepeat" user
        // default (converted from sixtieths of a second to seconds), but no more than
        two
        // seconds. This behavior is determined based on Inside Macintosh
        documentation
        // on the List Manager.
        NSUserDefaults *defaults
            = [NSUserDefaults standardUserDefaults];
        int keyThreshTicks
```

```
        = [defaults integerForKey:@"InitialKeyRepeat"];
        NSTimeInterval clearDelay
            = MIN(2.0/60.0*keyThreshTicks, 2.0);

        if (nil == mStringToFind)
        {
            mStringToFind = [[NSMutableString alloc] init];
            // lazily allocate the mutable string if needed.
        }
        [mStringToFind appendString:inString];

        // Cancel any previously queued future invocations
        [NSObject cancelPreviousPerformRequestsWithTarget:self
         selector:@selector(clearAccumulatingTypeahead)
         object:nil];

        // queue an invocation of clearAccumulatingTypeahead for the near future.
        [self performSelector:
         @selector(clearAccumulatingTypeahead)
         withObject:nil afterDelay:clearDelay];

        // Let the table's delegate do something with the string.
        // We use stringWithString to make an autoreleased copy so for its use,
        // since we may clear out the original string below before it can be used.
        [[self delegate] typeAheadString:
         [NSString stringWithString:mStringToFind]
         inTableView:self];
    }
}
```

clearAccumulatingTypeahead:
Clear out the string so our next typeahead will start from scratch.

```
- (void) clearAccumulatingTypeahead
{
    [mStringToFind setString:@""]; // clear out the queued string to find
}

@end
```

All that remains now is the nitty-gritty of finding the appropriate row to select based on the string the user has typed so far. Usually, this will mean finding a row that is a close match, not necessarily an exact match, to the search string. In a list of U.S. States, for example, "M" would select Maine; "MI" would select Michigan; "MIS" and "MISS" would select Mississippi; "MISSO" would select Missouri.

In our `TableTester` code, we select based upon whichever is the currently sorted column. We search linearly (ideally, it should be a binary search if the data set is large) for the row containing the dictionary with the value of the current sorting key that is a best match, using a case insensitive comparison. We use the last row if no match was found, e.g. if the user typed "Z" in a table with no "Z" entries. That row is selected and the table view scrolled to make that selection visible.

Listing 4: SortingDelegate.m

typeAheadString:inTableView:
Actually select the appropriate row based upon the string that has been typed.

```
(void) typeAheadString:(NSString *)inString
inTableView:(NSTableView *)inTableView
{
    NSTableColumn *col = [inTableView highlightedTableColumn];
    if (nil != col)
    {
        NSString *key = [col identifier];
        int i;
```

What's the point of driving a Jaguar if you can't get it out of first gear?

Mac OS X v10.2, aka Jaguar, is truly a marvel of innovation and software engineering. The latest release of the Mac OS includes over 150 new features such as the Point-to-Point Tunneling Protocol (PPTP), Lightweight Directory Access Protocol 3.0, integration of FreeBSD 4.4 and GCC 3.1 into Darwin, and Zero Configuration Networking (Rendezvous).

But having such a powerful machine on your desktop doesn't do much good if you don't know how to take full advantage of all that it has to offer.

Where the rubber meets the road.

In addition to covering the basics of Mac OS X, *Mac OS X: The Missing Manual*, 2nd Edition, has been fully updated to include all of Jaguar's new features. Serious power users and developers can get under the hood with chapters on connecting to Windows networks, wireless networking, and an update of open source technologies.

If you're new to Unix, or a Linux or Unix developer interested in the Mac OS, we've got everything you need to get into high gear. *Learning Unix for Mac OS X* teaches Mac developers how to use the Terminal application and the command interface and delves into several Unix applications. *Mac OS for Unix Developers*, written by two Unix developers deep into Mac OS X, shows Linux and Unix developers how to make the most of Jaguar as a development platform. And *Learning Cocoa with Objective C* is for anyone developing applications for Mac OS X. It's the only book on the topic that's been reviewed and approved by Apple's own engineers.

Go ahead. Start your engines.
We've got you—and Mac OS X—covered.

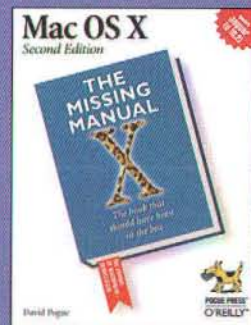


O'REILLY®

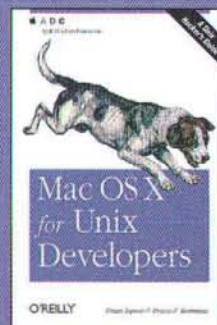
mac.devcenter.com

800-998-9938

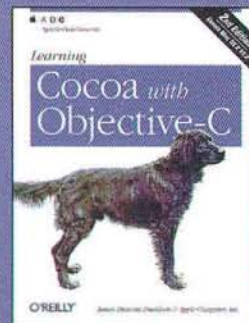
oreilly.com



**Mac OS X:
The Missing Manual, 2nd Edition**
David Pogue
ISBN 0-596-00450-8
\$29.95



Mac OS X for Unix Developers
Brian Jepson & Ernest E. Rothman
ISBN 0-596-00356-0
\$19.95



**Learning Cocoa
with Objective-C, 2nd Edition**
James Duncan Davidson
& Apple Computers, Inc.
ISBN 0-596-00301-3
\$34.95

©2002 O'Reilly & Associates, Inc.
O'Reilly is a registered trademark of O'Reilly & Associates Inc.
All other trademarks are property of their respective owners.

```

for ( i = 0 ; i < [oData count] ; i++ )
{
    NSDictionary *rowDict = [oData objectAtIndex:i];
    NSString *compareTo = [rowDict objectForKey:key];
    NSComparisonResult order
        = [NSString caseInsensitiveCompare:compareTo];
    if (order != NSOrderedDescending)
    {
        break;
    }
}
// Make sure we're not overflowing the row count.
if (i >= [oData count])
{
    i = [oData count] - 1;
}
// Now select row i — either the one we found, or the last row if not found.
[tableView selectRow:i byExtendingSelection:NO];
[tableView scrollRowToVisible:i];
}
}

```

One final note on typeahead: CodeWarrior has a nice feature of showing you the typeahead buffer so you can see what you have typed. In the TableTester program, but not listed here, are some minor additions to the TypeaheadTableView class that will display the current typeahead buffer if you have hooked up a text field to the appropriate outlet.

DISPLAYING MORE THAN JUST STRINGS

Usually, a table cell displays a piece of plain text, just an NSString returned in tableView:objectValueForTableColumn:row:. But this method is designed to return “id”, meaning any object. Without any extra effort, you can return an NSAttributedString or NSNumber instead of an NSString. And with just a little bit of setup, you can display other kinds of cells such as images and buttons.

First, you need to set the table’s columns to use a different cell type. In a convenient initialization method, such as your controller’s awakeFromNib method, just allocate a new cell object, such as an NSButtonCell or an NSImageCell. You may need to adjust attributes of your cell (for instance, setting a button cell’s type and image position). Then, find the NSTableColumn object corresponding to the column to affect, and replace the default text cell with your newly allocated instance, using setDataCell:. (Alternately, you can create a disembodied control in your nib file that represents the prototype cell and hook it all up in IB!)

In our TableTester program, we allocate a button cell and an image cell. The button cell is set up to be a checkbox (NSSwitchButton); the image cell doesn’t need any adjustment. Finally, the button and image cells are hooked up to the columns.

Listing 5: CellDelegate.m

```

// CellDelegate.m
// Set the table columns to use button and image cells rather than the default text cells.

- (void)awakeFromNib
{
    // Allocate the cells
    NSButtonCell *buttonCell
        = [[[NSButtonCell alloc] init] autorelease];

```

```

    NSImageCell *imageCell
        = [[[NSImageCell alloc] init] autorelease];

    // Find the columns
    NSTableColumn *buttonColumn
        = [tableView tableColumnWithIdentifier:@"checked"];
    NSTableColumn *imageColumn
        = [tableView tableColumnWithIdentifier:@"icon"];

    // Set up the button cell and install into the column
    [buttonCell setButtonType:NSSwitchButton];
    [buttonCell setImagePosition:NSImageOnly];
    [buttonCell setTitle:@""];
    [buttonColumn setDataCell:buttonCell];

    // Install the image cell
    [imageColumn setDataCell:imageCell];
}

```

To display these non-textual cells, you need to provide appropriate data in tableView:objectValueForTableColumn:row: and perhaps also make per-cell adjustments in tableView:willDisplayCell:forTableColumn:row:. In TableTester, we implement the data source and provide an appropriate object based on the given column. For the “icon” column, we provide an NSImage object based on the name in the data table. (This relies on their being an image available for the name; we have a few sample images created with Stick Software’s “Aquatint” program bundled with TableTester.) For the “checked” column, we provide an NSNumber object corresponding to the state of the checkbox. For the “name” column, we build up an attributed string to display in the default string cell. (And in the source code for TableTester, not shown here, we also have a cell for a “relevance” control, which uses a custom cell class.)

Listing 6: CellSource.m

```

// CellSource.m
// Provide the data for a given cell. We provide different data depending on which
// column is passed in as a parameter.

(id)tableView:(NSTableView *)tableView
objectValueForTableColumn:(NSTableColumn *)tableColumn
row:(int)rowIndex
{
    id result = nil;
    // Start out with the string that the SimpleSource returns from the dictionary
    id stringValue = [super tableView:tableView
objectValueForTableColumn:tableColumn row:rowIndex];

    // Now, handle special cases depending on the table column identifier.
    NSString *identifier = [tableColumn identifier];
    if ([identifier isEqualToString:@"icon"])
    {
        if (nil != stringValue)
        {
            result = [NSImage imageNamed:stringValue];
        }
    }
    else if ([identifier isEqualToString:@"checked"])
    {
        // Return NSNumber of 1 or 0. The line below builds the NSNumber
        // from the string or NSNumber currently in the dictionary.
        result = [NSNumber numberWithInt:[stringValue intValue]];
    }
    else if ([identifier isEqualToString:@"name"])
    {
        // Really, these dictionaries should be created once and cached....
        NSDictionary *plainAttr
            = [NSDictionary dictionaryWithObject:
                [NSFont systemFontOfSize:[NSFont systemFontOfSize]]];

```

The power of UNIX brought to the Mac

mac:ODBC

Open database connectivity
for the Macintosh

Advanced SQL Editor

Powerful multi-platform
and multi-DB SQL script editor

SQL Project

Multi-platform SQL
development environment

Team Diagram

Dynamic diagramming and charting
tool with CVS and database hooks

Data Architect

Multi-platform suite of database
modelling and design tools: includes
all the above software components

More Mac OS X products from
theKompany.com coming soon

**Take advantage of the power of UNIX
with theKompany.com's suite of database
software. Now also for Mac OS X.**

TheKompany.com, a UNIX software and services company, now offers
its powerful UNIX database connectivity tools for the Mac OS X
computing platform.

With theKompany.com's suite of database tools, you can do
just about anything with your SQL databases: connect to
SQL databases with mac:ODBC, create and edit SQL scripts
with the Advanced SQL Editor, create diagrams from CVS
trees and databases with Team Diagram, develop powerful
SQL-based projects with SQL Project and more.

Each of these tools is available separately, or as part of
Data Architect – theKompany.com's premiere database
modelling tool. All are offered with a unique open source
license, so you can tweak the program code to your liking.

Prices for these great products start at just **\$9.95** for
individual packages and top out at **\$49.95** for the complete
Data Architect suite.

For more information about pricing and availability, to try a
demo version, or to place your order online, visit us at
www.thekompany.com.

theKompany.com



```

        forKey:NSFontAttributeName];
    NSDictionary *boldAttr
    = [NSDictionary dictionaryWithObject:
        [NSFont boldSystemFontOfSize:[NSFont systemFontSize]]
        forKey:NSFontAttributeName];

    // Return an attributed string, with the first character boldface.
    result = [[[NSMutableAttributedString alloc] initWithString:stringValue
        attributes:boldAttr] autorelease];

    [result appendAttributedString:
        [[[NSAttributedString alloc] initWithString:[stringValue substringToIndex:1]
            attributes:boldAttr] autorelease]];
    [result appendAttributedString:
        [[[NSAttributedString alloc] initWithString:[stringValue substringFromIndex:1]
            attributes:plainAttr] autorelease]];
}
else
{
    // If not the special cases, we've gotten the string result from the superclass.
    result = stringValue;
}
return result;
}

```

SORTING TABLES

Displaying a table with its data sorted can enhance readability greatly. What's even more useful is if you give the user the ability to decide which column to sort a table by, and which direction to sort in. Astute readers will remember an article by Andrew Stone that covered sorting of tables in the August 2002 issue of MacTech; this section takes a slightly different approach (and offers Mac OS X 10.2 compatibility too).

To support column sorting, you need to implement `tableView:didClickTableColumn:` in your delegate. Your code would sort the data and redisplay it, highlighting the clicked column to provide feedback to the user as to what column the data is sorted by. If the user clicks on the sorted column a second time, the direction of the sort should change, and a small graphic in the column should indicate whether the table is sorted ascending or descending. For this to happen, you need to implement a bit of code.

First, let's deal with the actual sorting. `NSArray` and `NSMutableArray` provide a number of methods for sorting. The most convenient methods you can use are the methods `sortUsingFunction:` (if you have an `NSMutableArray`) or `sortedArrayUsingFunction:` (if you have an immutable `NSArray`). With these methods, you provide a C function that knows how to compare two objects and is given an arbitrary context for performing the sort. In our sample case, we use a simple structure specifying the key to sort upon, and the direction to sort in. Our function, `ORDER_BY_CONTEXT`, sets up the order for the comparison based upon the sorting direction, and then invokes either `caseInsensitiveCompare:` or `compare:` between the two objects. The former is useful for strings; the latter is useful for numbers or dates. Our method `sortData` invokes `sortUsingFunction:` on the data array using the current sorting key

and direction, then causes the table to redisplay itself with the `reloadData` method.

Listing 7: SortingDelegate.m

```

ORDER_BY_CONTEXT
C function to return the sort ordering for the given two objects and the given context.

typedef struct { NSString *key; BOOL descending; }
SortContext;

int ORDER_BY_CONTEXT (id left, id right, void *ctxt)
{
    SortContext *context = (SortContext*)ctxt;
    int order = 0;
    id key = context->key;
    if (0 != key)
    {
        id first, second; // the actual objects to compare

        if (context->descending)
        {
            first = [right objectForKey:key];
            second = [left objectForKey:key];
        }
        else
        {
            first = [left objectForKey:key];
            second = [right objectForKey:key];
        }

        if ([first respondsToSelector:
            @selector(caseInsensitiveCompare:)])
        {
            order = [first caseInsensitiveCompare:second];
        }
        else // sort numbers or dates
        {
            order = [(NSNumber *)first compare:second];
        }
    }
    return order;
}

sortData
Sort the data array based on the current sorting key (column) and sorting direction.

- (void) sortData
{
    SortContext ctxt={ mSortingKey, mSortDescending };
    [oData sortUsingFunction:ORDER_BY_CONTEXT context:&ctxt];
    [oTable reloadData];
}

```

Since we want to indicate the direction of sorting, we display a little triangle in the table column using the `-[NSTableView setIndicatorImage:inTableColumn:]` method. In Mac OS X 10.2, Apple provides official access to these images. But if you want your sorting-table application to work under 10.1, you have a couple of options. One is to include the images in your application's executable; another is to carefully make use of a private (undocumented) method in `NSTableView`. Using private methods is something you have to be very careful about, because Apple doesn't support them, and they could go away at any time. To make sure that the sample code will work on both 10.1 and 10.2, we test for compatibility using `respondToSelector:` to fail gracefully, then add class methods to `NSTableView` called `ascendingSortIndicator` and `descendingSortIndicator` to safely return the images we'll need.



REAL Software and MacTech present the REALbasic Showcase to highlight some of the fantastic solutions created by REALbasic users worldwide. The showcase illustrates the wide range of applications that developers using REALbasic can create. Some benefit any Mac user, and others are more specific. All of them are seriously cool!

REALbasic is the powerful, easy-to-use tool for creating your own software for Macintosh, Mac OS X, and Windows. It runs natively on Mac OS X as well as earlier versions of the Mac OS. For more information, please visit: <www.realbasic.com>.

The Made with REALbasic program is a cooperative effort between REALbasic users and REAL Software, Inc. to promote the products created using REALbasic and the people who create them. For more information about the Made with REALbasic program, please visit: <www.realbasic.com/realbasic/mwrb/Partners/MwRbProgram.html>.

Extend REALbasic with Advanced Plugins

Spreadsheet Controls:

We provide a wide range of spreadsheet controls for REALbasic, including multistyled and custom rendering spreadsheet controls.

A	B	C
This is some incredible list		
1	Some text	More text
2	Some text	More text
3	Some text	More text
4	Some text	More text
5	Some text	More text
6	Some text	More text
7	Some text	More text
8	Some text	More text

- More than 32k of rows.
- Classic, OS X and Win32.
- Accelerated for maximum speed.
- Images in cells.



Cryptography:

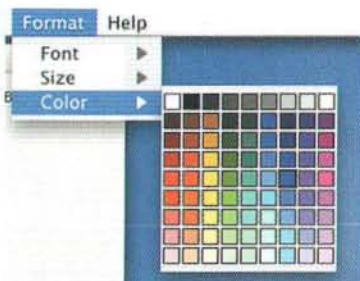
We provide data encryption, encoding, compression and hashing for REALbasic.

Supported Algorithms:

- Encryption:
 - e-CryptIt
 - BlowFish (448 bit)
 - AES (Rijndael) (256 bit)
- Encoding:
 - e-CryptIt Flexible
 - Base 64
 - BinHex
 - MacBinary III
 - AppleSingle / Double
 - UUcoding
- Compression:
 - Zip on Strings and streams (.gz)
- Hashing and Checksums:
 - CRC32 / Adler32
 - MD5 / HMAC_MD5
 - SHA / SHA1 / HMAC_SHA1

Other Plugins:

We have many other plugins for REALbasic, including plugins to do advanced MacOS Toolbox tasks and more custom Controls.



Speed up development and make more advanced applications by using plugins ! Get free demos at www.einhugur.com



Einhugur Software
sales@einhugur.com
www.einhugur.com



piPop

Pop-up Hierarchical
File Navigation and Launcher



TelnetLauncher

Bookmark and Launch your
Telnet and SSH sessions



SimpleKeys

Set your Function Keys
to type stuff for you!

piDog Software

<http://www.pidog.com/>

Whistle Blower

Enterprise server monitor and restart utility
whistleblower.sentman.com

Connect to and validate the response from web servers, cgi scripts and over 23 other types of servers.

Send email, pages and perform unattended restarts via MasterSwitch or PowerKey.

Shifts make sure that the person on call when the server goes down is the one who gets the page.

68k, PPC and Carbon

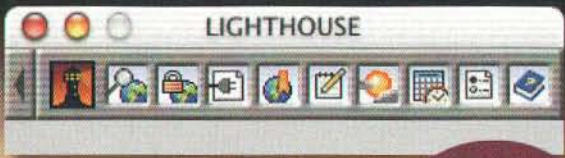
Web based administration lets you check on and restart your servers from anywhere.

Customize your response to an outage with Apple Script.

email us at whistleblower@sentman.com

LIGHTHOUSE

Get Organized with this Time-Saving Internet Assistant



- Search the Web with Free Plugins
- Plugins Manager/Editor
- Local Weather for 160+ Cities
- Calendar Saves Your Important Dates & Events
- World Clock Displays Local Time for 65 Cities
- Note Keeper Saves Code Snippets, Ideas, To Do Lists...
- Store Account Info for Membership Sites & Web Mail
- Enhanced Web Favorites Organizer
- Export Your Personal Data to Text, XML & HTML
- Now Available for Classic Mac & Mac OS X

NEW! Save Calendars as HTML Pages

DOWNLOAD NOW
www.ebutterfly.com

electric butterfly

Corona

accounting software



"Easy to set up, easy to use, and excellent support from the developer."

Five stars on VersionTracker.com
 Four cows on Tucows.com

- cash entry
- invoicing
- payroll
- reports

Version 1.9 now with: sales tax accounting
 "keyless entry" invoices
 drag 'n drop accounts

\$64.95

Free 30-day trial
<http://homepage.mac.com/idlewild/CoronaUS.hqx>



A best friend for business!

p.o. box 472 • aurora • oregon • 97002
idlewild@mac.com

iScreensaver Designer

the cross-platform solution for Macintosh and Windows



Version 3.0 featuring **OS X** coming soon!



Mac designers build professional screen savers without touching a Windows machine!

- build both Macintosh and Windows screensavers with a single click, no matter what system* you are using!
- use any QuickTime 6.0 movie format: Macromedia Flash 5.0, MPEG, Cinepak, MP3, Midi, AVI, DV Video... or, now with version 3.0, build your own basic Slide Shows!
- include a hidden movie that can be unlocked with a registration code
- customize and fully-brand both Installers and Screensaver control panels with pictures and text
- Screensavers install without DLLs, extensions, or restarts

simple WYSIWYG editor

supports interactive Flash and QuickTime

consistent cross-platform user interface

try before you buy
 fully functional
 online downloads



the iScreensaver Designer editing environment

creating screensavers for both Windows and Macintosh has never been this easy

<http://iScreensaver.net>
 email: info@iScreensaver.net

* supported systems, as of June 2002, include:
 Macintosh OS 8.6 to 9.2.2, Microsoft Windows 95/98/ME, NT4/NT2000/XP
 iScreensaver Designer version 3.0 will support up to Macintosh OS 10.2

©2000-2002 Xochi Media Inc. Made using REALbasic.

Listing 8: NSTableView+util.m

ascendingSortIndicator
Safely return the ascending sort triangle image; works on both 10.1 and 10.2.

```
+ (NSImage *) ascendingSortIndicator
{
    NSImage *result
    = [NSImage imageNamed:@"NSAscendingSortIndicator"];
    if (nil == result
        && [[NSTableView class] respondsToSelector:
            @selector(_defaultTableHeaderSortImage)])
    {
        result = [NSTableView _defaultTableHeaderSortImage];
    }
    return result;
}
```

descendingSortIndicator
Safely return the descending sort triangle image; works on both 10.1 and 10.2.

```
+ (NSImage *) descendingSortIndicator
{
    NSImage *result
    = [NSImage imageNamed:@"NSDescendingSortIndicator"];
    if (nil == result
        && [[NSTableView class] respondsToSelector:
            @selector(_defaultTableHeaderReverseSortImage)])
    {
        result
        = [NSTableView _defaultTableHeaderReverseSortImage];
    }
    return result;
}
```

Now to specify how clicking on a column sorts by that column. If it's a click on an already selected column, we reverse the sorting direction. Our method `sortByColumn:` is fairly straightforward; given a table column, we switch sort order if it's a click on the previously saved column; otherwise we change to a new sorting column. We invoke the column sorting method in the delegate method `tableView: didClickTableColumn:` to respond to column clicks. In a full application, you may want to store a preference of the sorted column and initially sort appropriately, perhaps calling `sortByColumn:` in your `awakeFromNib` method.

Listing 9: SortingDelegate.m

sortByColumn:
Sort the table by the given column, changing sort direction if already sorted by that column.

```
- (void)sortByColumn:(NSTableColumn *)inTableColumn
{
    if (mSortingColumn == inTableColumn)
    {
        // User clicked same column, change sort order
        mSortDescending = !mSortDescending;
    }
    else
    {
        // User clicked new column, change old/new column headers,
        // save new sorting column, and re-sort the array.
        mSortDescending = NO;
        if (nil != mSortingColumn)
        {
            [oTable setIndicatorImage:nil
                inTableColumn: mSortingColumn];
        }
        [self setSortingKey:[inTableColumn identifier]];
        [self setSortingColumn:inTableColumn];
        [oTable setHighlightedTableColumn:inTableColumn];
    }
}
```

```

{
    [oTable setIndicatorImage: (mSortDescending
        ? [NSTableView descendingSortIndicator]
        : [NSTableView ascendingSortIndicator])
        inTableColumn: inTableColumn];
    // Actually sort the data
    [self sortData];
}
```

tableView: didClickTableColumn:
User clicked on a table column, so sort (or invert the sort) by that column.

```
(void)tableView:(NSTableView *)inTableView
didClickTableColumn:(NSTableColumn *)inTableColumn
{
    [self sortByColumn:inTableColumn];
}
```

MAINTAINING SELECTION FOR A SORT

In the above example, one potential problem is what will happen if any rows are selected when you sort the table. The selected rows will remain the same positionally, but the items selected will no longer be the same. You could clear out any selection when you sort, but it's more useful to maintain the selected items through a sort. The method `saveSelectionFromTable:` gathers up the array items into an `NSSet`, and `restoreSelection: toTable:` finds those items after the array has been sorted and reselects them. These methods should handle simple cases, though the selection restoration code may not perform well for large arrays since `-[NSArray indexOfObjectIdenticalTo:]` is essentially a linear search.

Listing 10: SortingDelegate.m

saveSelectionFromTable:
Create a set that represents the current selection from a table, for later restore.

```
- (NSSet *) saveSelectionFromTable:(NSTableView *)inTableView
{
    NSMutableSet *result = [NSMutableSet set];
    NSEnumerator *theEnum
    = [inTableView selectedRowEnumerator];
    NSInteger *rowNum;

    while (nil != (rowNum = [theEnum nextObject]))
    {
        id item = [oData objectAtIndex:[rowNum intValue]];
        [result addObject:item];
    }
    return result;
}
```

restoreSelection: toTable:
Restore the selection from the given set of row objects after a table has been sorted.

```
- (void) restoreSelection:(NSSet *)inSelectedItemNums
toTable:(NSTableView *)inTableView
{
    NSEnumerator *theEnum
    = [inSelectedItemNums objectEnumerator];
    id item;
    NSInteger savedLastRow;

    [inTableView deselectAll:nil];

    while (nil != (item = [theEnum nextObject]))
    {
        NSInteger row = [oData indexOfObjectIdenticalTo:item];
    }
}
```



Save 20%
off these titles at

BORDERS®

Sale begins
November 3, 2002

Visual QuickStart Guides

Get up and running quickly!

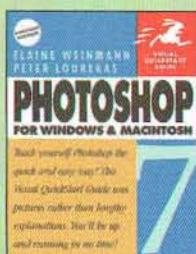
Our *HTML for the World Wide Web, Fifth Edition, with XHTML and CSS: Visual QuickStart Guide* is better than ever...now in its 5th Edition!



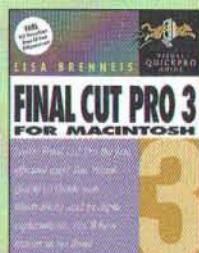
HTML for the World Wide Web, Fifth Edition, with XHTML and CSS: Visual QuickStart Guide
Elizabeth Castro
0-321-13007-3 • \$21.99

The Web is changing how it does business and so should you! If you're still coding like it was 1999, you need to update your HTML skills with Elizabeth Castro's *HTML for the World Wide Web, Fifth Edition, with XHTML and CSS: Visual QuickStart Guide*. This latest edition of the original book on HTML will have you creating complex, dynamic sites that look good across all browsers and platforms—including hand-held devices and cell phones—in no time!

More guides to get you up and running quickly!



Photoshop 7 for Windows and Macintosh: Visual QuickStart Guide
Elaine Weinmann and Peter Lourekas
0-201-88284-1 • \$24.99



Final Cut Pro 3 for Macintosh: Visual QuickPro Guide
Lisa Brenneis
0-321-11583-X • \$29.99



Macromedia Dreamweaver MX for Windows and Macintosh: Visual QuickStart Guide
J. Tarin Towers
0-201-84445-1 • \$24.99



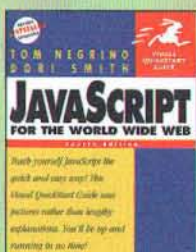
Macromedia Flash MX for Windows and Macintosh: Visual QuickStart Guide
Katherine Ulrich
0-201-79481-0 • \$24.99



Mac OS X 10.2: Visual QuickStart Guide
Maria Langer
0-321-15801-6 • \$21.99



iPhoto 1.1 for Mac OS X: Visual QuickStart Guide
Adam Engst
0-321-12165-1 • \$19.99



JavaScript for the World Wide Web, 4th Edition: Visual QuickStart Guide
Tom Negrino and Dori Smith
0-201-73517-2 • \$19.99



Macromedia Flash MX Advanced for Windows and Macintosh: Visual QuickPro Guide
Russell Chun
0-201-75846-6 • \$29.99

```

// look for an exact match, which is OK here.
if (NSNotFound != row)
{
    [tableView selectRow:row byExtendingSelection:YES];
    savedLastRow = row;
}
[tableView scrollRowToVisible:savedLastRow];
}

```

To make use of these selection methods, we just save the selection before sorting and restore it afterwards. This is the new implementation of `sortData` from above.

Listing 11: SortingDelegate.m

```

sortData
Sort the data, but this time, save the selection before the sort and restore it afterwards.

- (void) sortData
{
    SortContext ctxt={ mSortingKey, mSortDescending };
    NSSet *oldSelection = [self saveSelectionFromTable:oTable];

    // sort the NSMutableArray
    [oData sortUsingFunction: ORDER_BY_CONTEXT context:&ctxt];

    [oTable reloadData];
    [self restoreSelection:oldSelection toTable:oTable];
}

```

DRAG AND DROP TO REARRANGE ROWS

Some tables benefit from the ability to drag rows around to reorder the table's contents. For instance, the "International" panel of the System Preferences allows you to specify the languages you prefer to use when using the Mac.

According to the documentation on the `NSTableDataSource` informal protocol, there are three methods you would implement in your table's data source to facilitate drag and drop. One is for grabbing the data when you drag; one validates whether data can be dropped on your table; one performs the drop. It sounds simple, but there are always subtleties to work through.

If your table is going to handle drag and drop, you need to decide what operations that entail. In this segment, we are merely rearranging rows in the table, then it is simplest to merely keep track of the row index (or indexes) being dragged, so that your code can directly manipulate the data array's ordering.

The first of the three methods, `tableView: writeRows: toPasteboard:` is tasked with putting data corresponding to the given rows that are being dragged into a pasteboard. In Cocoa, there are multiple pasteboards available, and each pasteboard can contain multiple kinds of data. In our case, we put a representation of which row indexes were dragged, for the purposes of mere row reordering. This is identified as a constant string, "MyRowListPasteboardType", identified in the code as `kPrivateRowPBType`. (If your program were to have multiple table views — where it would be possible to drag from one table to the other — you'd have to take care to

specify separate pasteboard data types, and/or encode the source of the row indexes, so you wouldn't inadvertently mix apples and oranges.)

To encode is the list of dragged row indexes, we just use the `NSArchiver` class to turn the `NSArray` we are handed containing all of the row indexes into a single `NSData` object.

Listing 12: DraggableSource.m

```

tableView: writeRows: toPasteboard:
Put a list of the given rows onto a private pasteboard.

- (BOOL)tableView:(NSTableView *)tableView
writeRows:(NSArray *)inRows
toPasteboard:(NSPasteboard *)inPasteboard
{
    NSData *archivedRowData
    = [NSArchiver archivedDataWithRootObject:inRows];
    [inPasteboard declareTypes:
     [NSArray arrayWithObjects:kPrivateRowPBType, nil]
     owner:nil];
    [inPasteboard setData: archivedRowData
     forType:kPrivateRowPBType];
    return YES;
}

```

The next method, `tableView: validateDrop: proposedRow: proposedDropOperation:`, is needed to validate whether a drop can occur; this is called repeatedly as the user drags over the table view. Tables can accept drops onto a row, or between rows; in our case, we only want to accept drops between rows (the given `NSTableViewDropOperation` must be `NSTableViewDropAbove`), and we only want to accept the drop if the clipboard has our own private data type.

Listing 13: DraggableSource.m

```

tableView: validateDrop: proposedRow: proposedDropOperation:
Determine whether a drop can take place, and how it will be treated.

(NSDragOperation)tableView:(NSTableView *)tableView
validateDrop:(id <NSDraggingInfo>)inInfo
proposedRow:(int)inRow
proposedDropOperation:
(NSTableViewDropOperation)inOperation
{
    // Look for our private type for reordering rows.
    NSString *type
    = [[inInfo draggingPasteboard]
     availableTypeFromArray:[NSArray
     arrayWithObjects:kPrivateRowPBType, nil]];

    if (inOperation == NSTableViewDropAbove
     && [type isEqualToString:kPrivateRowPBType])
    {
        return NSDragOperationMove;
    }
    return NSDragOperationNone;
}

```

The final method, `tableView: acceptDrop: row: dropOperation:` actually performs the drop. This method checks the pasteboard type available, and if it is our private identifier representing row indexes, it performs the data rearrangement. It works by copying out the appropriate rows of data from our data array, replacing them with special `NSNull` values; then it inserts these

data rows into the table; finally it compacts the table by removing the NSNull placeholders.

Listing 14: DraggableSource.m

```
tableView:acceptDrop:row:dropOperation:
Handle a drop. If it's our private pasteboard listing the rows to move, actually move
the data.

- (BOOL)tableView:(NSTableView*)tableView
  acceptDrop:(id <NSDraggingInfo>)inInfo
  row:(int)inRow
  dropOperation:(NSTableViewDropOperation)inOperation
{
    // Look for our private type for reordering rows.
    NSString *type = [[inInfo draggingPasteboard]
        availableTypeFromArray:[NSArray
            arrayWithObjects:kPrivateRowPType, nil]];
    if ([type isEqualToString:kPrivateRowPType])
    {
        NSData *archivedRowData
            = [[inInfo draggingPasteboard]
                dataForType:kPrivateRowPType];
        NSArray *rows = [NSUnarchiver
            unarchiveObjectWithData:archivedRowData];
        NSMutableArray *movedRows
            = [NSMutableArray arrayWithCapacity:[rows count]];
        NSEnumerator *theEnum = [rows objectEnumerator];
        id theRowNumber;

        // First collect up all the selected rows, then put null where it was in the array
        while (nil != (theRowNumber = [theEnum nextObject]))
        {
            int row = [theRowNumber intValue];
            [movedRows addObject:[oData objectAtIndex:row]];
            [oData replaceObjectAtIndex:row
                withObject:[NSNull null]];
        }

        // Then insert these data rows into the array
        [oData replaceObjectsInRange:NSMakeRange(inRow, 0)
            withObjectsFromArray:movedRows];

        // Now, remove the NSNull placeholders
        [oData removeObjectIdenticalTo:[NSNull null]];

        // And refresh the table. (Ideally, we should turn off any column highlighting)
        [tableView deselectAll:nil];
        [tableView reloadData];
    }
    return YES;
}
```

One last step remains. The table view must register itself in being interested in the pasteboard type representing our row indexes; a good place to perform this is in your `awakeFromNib` method.

```
[oTable registerForDraggedTypes:
    [NSArray arrayWithObjects:kPrivateRowPType, nil]];

```

DAG AND DROP OF DATA

It can also be useful to drag data from the tables into other applications, or import data via drag and drop. In this segment, we'll explore data export via drag and drop, leaving importing of data as another proverbial "exercise for the reader."

For dragging data out, only the first method, `tableView:writeRows:toPasteboard:`, is affected, as it packages up the data to export to another program. (If you were to allow dropping onto your table views from external sources, such as cells

dropped in from a spreadsheet, you would want to modify `tableView:validateDrop:proposedRow:proposedDropOperation:` and `tableView:acceptDrop:row:dropOperation:` to accept other types of pasteboard data, and register for those pasteboard types in your `awakeFromNib` method.)

Our `TableTester` program adds two additional pasteboard types to the pasteboard in addition to the private type for rearranging rows: `NSStringPboardType` (generic text) and `NSTabularTextPboardType` (tabular text, as for a spreadsheet).

To build up the strings, we enumerate through each of the rows; for each row, we enumerate through all the columns. Each row of data fills up the buffer with strings for each cell, separated by a tab or a newline, with a final extra newline after each row. If your application needed the data formatted differently (for instance, always separated by tabs, or always in a specific column ordering regardless of the current display, or as rich text), you would modify this code to build the appropriately structured data.

Listing 15: DraggableSource.m

```
tableView:writeRows:toPasteboard:
Write the text data on the given rows onto the pasteboard.

(BOOL)tableView:(NSTableView *)tableView
  writeRows:(NSArray*)inRows
  toPasteboard:(NSPasteboard*)inPasteboard
{
    NSData *archivedRowData
        = [NSArchiver archivedDataWithRootObject:inRows];

    NSArray *tableColumns = [tableView tableColumns];
    NSMutableString *tabsBuf = [NSMutableString string];
    NSMutableString *textBuf = [NSMutableString string];
    NSEnumerator *rowEnum = [inRows objectEnumerator];
    NSInteger *rowNumber;

    while (nil != (rowNumber = [rowEnum nextObject]))
    {
        int row = [rowNumber intValue];
        NSEnumerator *colEnum = [tableColumns objectEnumerator];
        NSTableColumn *col;
        while (nil != (col = [colEnum nextObject]))
        {
            id columnValue
                = [self tableView:tableView
                    objectValueForTableColumn:col row:row];
            NSString *columnString = @"";
            if (nil != columnValue)
            {
                columnString = [columnValue description];
            }
            [tabsBuf appendFormat:@"%t", columnString];
            if (![columnString isEqualToString:@""])
            {
                [textBuf appendFormat:@"%n", columnString];
            }
        }
        // delete the last tab. (But don't delete the last CR)
        if ([tabsBuf length])
        {
            [tabsBuf deleteCharactersInRange:
                NSMakeRange([tabsBuf length]-1, 1)];
        }
        // Append newlines to both tabular and newline data
        [tabsBuf appendString:@"\n"];
        [textBuf appendString:@"\n"];
    }
}
```

```

// Delete the final newlines from the text and tabs buf.
if ([tabsBuf length])
{
    [tabsBuf deleteCharactersInRange:
        NSRange([tabsBuf length]-1, 1)];
}
if ([textBuf length])
{
    [textBuf deleteCharactersInRange:
        NSRange([textBuf length]-1, 1)];
}

// Set up the pasteboard
[inPasteboard declareTypes:
    [NSArray arrayWithObjects: NSTabularTextPboardType,
        NSStringPboardType, kPrivateRowPboardType, nil]
    owner:nil];

// Put the data into the pasteboard for our various types
[inPasteboard setString:[NSString stringWithString:textBuf]
    forType:NSStringPboardType];
[inPasteboard setString:[NSString stringWithString:tabsBuf]
    forType:NSTabularTextPboardType];
[inPasteboard setData: archivedRowData forType:
    kPrivateRowPboardType];

return YES;
}

```

One obscure trick remains. In order for you to be able to drag data outside of your application, you need to override a method in `NSTableView`. Your table view must therefore be of a custom class. If you don't override this method, you will not be able to drag data out of a table into another application! Hopefully Apple will fix this in a future version of Mac OS X.

Listing 16: TypeaheadTableView.m

```

draggingSourceOperationMaskForLocal:
Allow drags outside of an application.

- (NSDragOperation)draggingSourceOperationMaskForLocal:
    (BOOL)isLocal
{
    if (isLocal) return NSDragOperationEvery;
    else return NSDragOperationCopy;
}

```

COPYING ROWS TO THE CLIPBOARD

With drag and drop supported, it's actually quite easy to add the capability to copy selected rows of a table to the clipboard. We employ the method `tableView: writeRows: toPasteboard:`, passing in the general pasteboard, to respond to the Copy menu item. We check to make sure that the table's data source implements that method, so this method will fail gracefully if the current table doesn't support the clipboard.

Listing 17: AppController.m

```

copy:
Handle the request to copy rows from the table.

- (IBAction) copy:(id)sender
{
    // Get the NSTableView we want to copy from. In this case, we determine the

```

```

// "current" table view by getting the tab view item's initialFirstResponder,
// which is set in the nib.
id currentTable
    = [[oTabView selectedTabViewItem] initialFirstResponder];

// Now put the selected rows in the general pasteboard.
if ([currentTable dataSource] respondsToSelector:
    @selector(tableView:writeRows:toPasteboard:))
{
    (void) [[currentTable dataSource] tableView:currentTable
        writeRows:[currentTable selectedRows]
        toPasteboard:[NSPasteboard generalPasteboard]];
}
}

```

It's actually that simple! All that is missing is a method in `NSTableView` called `selectedRows`, to return an array of row indexes. This is fixed quickly by adding a category method to `NSTableView`.

Listing 18: NSTableView+util.m

```

selectedRows
Return an array of the selected row numbers of the table.

- (NSArray *) selectedRows
{
    NSEnumerator *theEnum = [self selectedRowEnumerator];
    NSInteger *rowNumber;
    NSMutableArray *rowNumberArray = [NSMutableArray
        arrayWithCapacity:[self numberOfSelectedRows]];

    while (nil != (rowNumber = [theEnum nextObject]))
    {
        [rowNumberArray addObject:rowNumber];
    }
    return rowNumberArray;
}

```

UNTIL WE MEET AGAIN

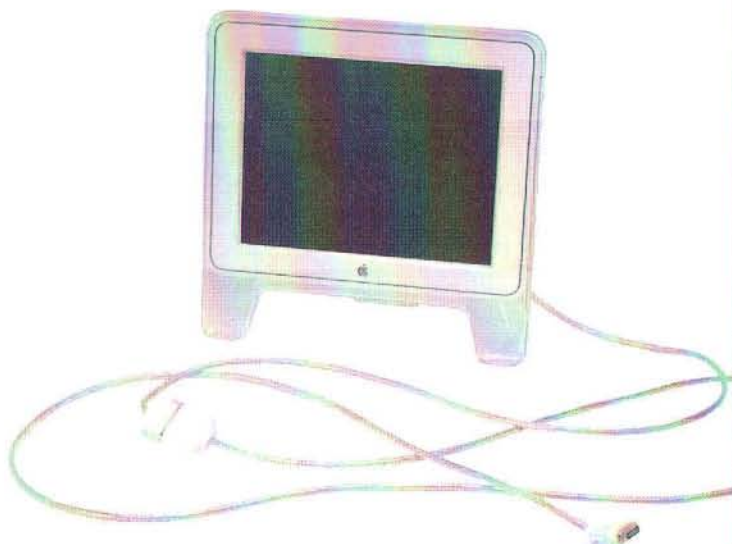
If you've made to the end of part two, congratulations — you can go forth and create some amazingly rich table interfaces. But there are *still* more cool things you can do with `NSTableView`, and this is why there's another part in the series on the way. Tune in next month for part three, in which we'll some advanced techniques, including the technique for correctly displaying those trendy striped tables that you see in the "iApps," and a subclass that merges certain cells together into wider cells.



ADE
Extension

Place the display up to 15 feet away from the CPU without any loss of quality.

Extend the cable on your 15", 17", 22" or 23" Apple display by 3 meters (10 ft).



**CoolMac™
Keyboard**

This silicone-cast keyboard is completely rollable and liquid resistant, making it perfect for travel or harsh conditions.

The CoolMacKeyboard is the world's most flexible keyboard, in more ways than one.



Dr. Bott's complete focus is serving resellers with hundreds of products from over 30 manufacturers. We scour the globe to bring you the best Mac products. New resellers and manufacturers are always welcome.

503.582.9944

www.drbott.com

877.611.2688 (Toll Free)

By Michael R. Harvey

Star Wars: Galactic Battlegrounds

Ed. Note - This is only going to hurt for a second, then a little pressure, then it's going to hurt really bad, so just try to breathe. You might smell a burning smell, just ignore that.

What's this? A game review in MacTech? Isn't that one of the signs of the Apocalypse? Actually, no. Doing an entire games issue would be, and we aren't there, at least not yet.

No, this is more of an intervention than the end of the world (although it might feel like it at first). Have you looked in a mirror lately? You're pale, your posture has gone to hell, and the IV of Dr. Pepper isn't helping your complexion out. It's high time you stopped coding for a bit, and do something else with all the processing power at your fingers. Something frivolous. Notice we aren't recommending you actually turn the computer off, and go outside for some sun and fresh air. We can't expect miracles right out of the gate. We'll start with baby steps. You can still have the bad posture, and the Mountain Dew injections, but let's try some completely wasted fun time. Let's play a game. And why not play one ported by those geniuses at Westlake Interactive and published by the folks at Aspyr, the company that has published some of the best titles the Mac platform has seen the past several years.

Star Wars: Galactic Battlegrounds is a recent release from Aspyr. As you might have guessed by now, it is a game from the LucasArts guys centering around everyone's favorite space opera, Star Wars. This game is from the real time strategy category, similar to Command and Conquer or StarCraft. Built on the Age of Empires 2 engine, you are able to take command of one of six different factions, from the evil Galactic Empire, to the Naboo, to Wookiees. You are then tasked with missions, and complete them through managing your resources, trading with your allies, building your forces and defeating the enemy. Good times!

The system requirements for the game are actually quite tame. A G3 processor, 64 MB RAM, a video card capable of displaying 256 colors. Not too taxing for almost anyone, is it? Network play can be done over a LAN, or even with a 28.8 modem (although more is always better when it comes to gaming, as it is with most things). The graphics look great. Units, and structures are rendered with good detail. It is easy to discern who is who. The sound is quite nice, as well. You can see, and hear, the effort that went into the details to make the Star Wars universe come alive on your screen.

If you are already familiar with Age of Empires 2, this article is not for you, go outside already. If you're still with me, and have played Age of Empires 2, you are pretty much ready to go. Galactic Battlegrounds uses many of the same interface controls and command keys. As with Age, having a multi-button mouse really helps out. That right button gets a work out in this game. Control clicking so often gets to be a drag, and we just can't have that.



If you are unfamiliar with this type of gaming, you might want to start out with the Basic Training missions. In these, you will be led by Qui-Gon Jinn in how to establish your base, gather resources, build units and structures, and conduct battle. It's not absolutely necessary, though. The controls really are quite simple, so you can jump right into the game if you wish.

Game play revolves around either single player or multi-player use.

In single player mode, there are a few options for game play. You can choose to play the Campaigns. These are a series of missions you are tasked to complete. You become one of several characters from the Star Wars universe. The campaigns take you through much of the mythos of Star Wars, not only the movies but the novels, as well.

You also have the option of playing stand alone missions. In these games, you set the parameters, such as number of opponents, type of map, and starting resources. The game then

puts you on a map, and you fight it out based on the rules you established. Included as part of the game is a scenario editor in which you can create your own maps, and play conditions. These can be shared with other gamers, as well as you being able to play scenarios created by other users.

There are two ways to engage multiplayer action. One is where the host sets up the games rules, and other gamers join in. The others access the hosts game either over a LAN, or over the internet. All connections are via TCP/IP. If the gamers are on the hosts LAN, they will see the game automatically. Users coming in over the Internet will need to know the IP address of the host to join the game.

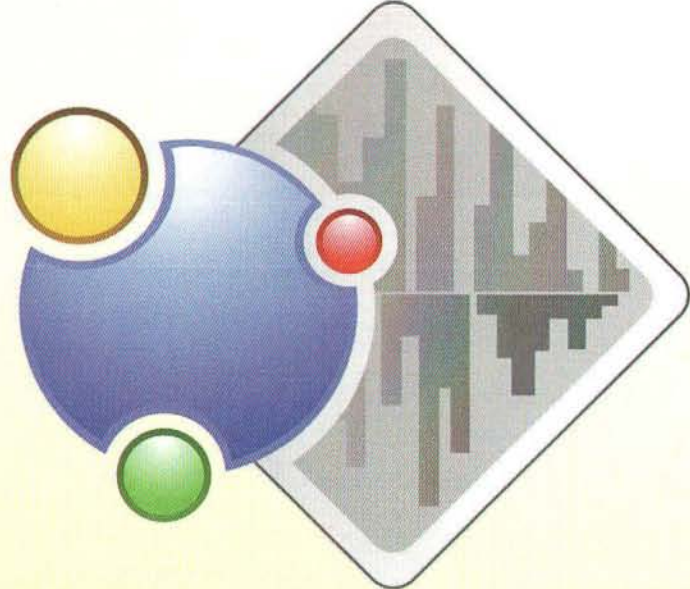
The other method of play in Galactic Battlegrounds is via Game Ranger. Game Ranger is a freeware application that helps you link up with other Mac players, and set up online games. Recently, Scott Kevil, Game Ranger's author, released a public beta of version 3.0 which adds support for OS X. If you do not want to load up beta software, you will need to boot into OS 9 to play online via the current version of Game Ranger. However you access it, Game Ranger is a very worthwhile tool to have for gaming. It currently supports over 50 game titles, and is Mac only.

This game is a great ride, and wretchedly addictive. The many detailed options you can change and tweak will allow you a vast array of game possibilities. You will definitely waste a goodly amount of time on this title. Plan accordingly. The only drawback you may run into is with the difficulty settings. Easy is too easy. There isn't much challenge. The enemy just waits for you to come mop the floor with them. Jumping from there to normal could be a shock, as the normal setting will smack you around handily if you're not ready for it. Hard is, of course, hard, but it keeps your pulse rate up.

It is unfortunate that so many of the games created around the Star Wars universe don't make it to the Macintosh platform. It is always a treat when companies like Westlake and Aspyr secure the rights to one of the many great titles LucasArts produces. Galactic Battlegrounds is no exception. Hopefully we will get to see more of these games in the near future. And in fact, we will very soon. Star Wars Jedi Knight II: Jedi Outcast is currently in development for release later this year. Galactic Battlegrounds has an MSRP of \$50, and is available from almost anywhere Mac software is sold.

One last note. Kudos to Aspyr for their packaging of Galactic Battlegrounds. The box is small, and compact; just big enough to hold the CD, manual, and technology chart. Not one of these 47 pound monstrosities that hold one CD. They saved a tree or two. Good for them.

<http://www.aspyr.com/>
<http://www.gameranger.com/>



IPNetMonitorX

- **The Swiss Army Knife of Mac OS X Internet Diagnostic Tools**
- **15 Fully Integrated Tools—Monitor, Link Rate, Address Scan, TCP Dump and more**
- **Troubleshoot and Solve Network Problems**
- **Responsive, Intuitive, Easy to Use Interface**
- **Fast Multi-Threaded Architecture—see network behavior as it happens**
- **21-Day Fully Functional Free Trial**



Download Your Copy Now at

www.sustworks.com/mac

SUSTAINABLE
Softw*orks*
Tools for Internet Travel



List of Advertisers

AEC Software.....	49
Aladdin Knowledge Systems, Inc.....	35
Aladdin Systems, Inc.....	37
Avesta Computer Services, Ltd.....	23
Big Nerd Ranch, Inc.....	20
Borland Software Corporation.....	59
Computer Systems Odessa, Corp.....	31
DevDepot.....	28-29
Dr. Bott LLC.....	77
Einhugur Programming Resources.....	70
Electric Butterfly.....	71
Eudora Internet Mail Server.....	56
Exabyte.....	9
FairCom Corporation.....	1
Felt Tip Software.....	23
Fetch Softworks.....	21
Flickinger Software.....	71
Full Spectrum Software, Inc.....	19
IDG World Expo Corporation.....	50-51
InformINIT.com.....	56
James Sentman Software.....	70
Kaidan.....	45
Karelia Software.....	57
Key 3 Media / COMDEX.....	15
Lemke Software.....	56
Lingo Systems.....	53
MacDirectory.....	25
Mathemaesthetics, Inc.....	47
MYOB US, Inc.....	63
/n software inc.....	7
Netopia, Inc.....	43
O'Reilly & Associates, Inc.....	65
OpenBase International, Ltd.....	61
Paradigma Software.....	27
Peachpit Press.....	73
Perforce Software, Inc.....	81
piDog Software.....	70
PrimeBase (SNAP Innovation).....	11
REAL Software, Inc.....	69-71
REAL Software, Inc.....	82
Runtime Revolution Limited.....	ifc2
Sig Software.....	56
Small Dog Electronics.....	17
Stone Design Corp.....	52
SuSE Inc.....	54
Sustainable Softworks.....	79
SyBase, Inc.....	2-3
The Software MacKiev Company.....	60
theKompany.com.....	67
Thursby Software Systems, Inc.....	13
TLA Systems Ltd.....	57
Trapcode Software.....	57
Trinifinity Software.....	56
Utilities4Less.com.....	44
WIBU-SYSTEMS AG.....	33
Xochi Media Inc.....	71
Xplain Corporation.....	39
Zero G Software.....	41

List of Products

Accessories • DevDepot.....	28-29
Accessories • Dr. Bott LLC.....	77
Adobe Press • Peachpit Press.....	73
Assorted Utilities • theKompany.com.....	67
Big Nerd Ranch • Big Nerd Ranch, Inc.....	20
Books • O'Reilly & Associates, Inc.....	65
c-tree Plus • FairCom Corporation.....	1
COMDEX • Key 3 Media / COMDEX.....	15
Concept Draw • Computer Systems Odessa, Corp.....	31
Consulting & Training Services • Avesta Computer Services, Ltd.....	23
Corona • Flickinger Software.....	71
DAVE • Thursby Software Systems, Inc.....	13
Development & Testing • Full Spectrum Software, Inc.....	19
DragThing • TLA Systems Ltd.....	57
EIMS • Eudora Internet Mail Server.....	56
FastTrack • AEC Software.....	49
Fetch • Fetch Softworks.....	21
GraphicConverter • Lemke Software.....	56
InformINIT.com • InformINIT.com.....	56
InstallAnywhere • Zero G Software.....	41
InstallerMaker, StuffIt • Aladdin Systems, Inc.....	37
IP®Works! • /n software inc.....	7
IPNetRouter & IPNetSentry • Sustainable Softworks.....	79
iScreensaver Designer • Xochi Media Inc.....	71
JBuilder • Borland Software Corporation.....	59
Long Distance Phone Service • Utilities4Less.com.....	44
MacDirectory • MacDirectory.....	25
Macworld Conference & Expo • IDG World Expo Corporation.....	50-51
Macworld Special Interest Pavilions • Xplain Corporation.....	39
MYOB • MYOB US, Inc.....	63
OpenBase • OpenBase International, Ltd.....	61
Photographic VR Solutions • Kaidan.....	45
piDog Utilities • piDog Software.....	70
PrimeBase • PrimeBase (SNAP Innovation).....	11
QuickerHelp & Consulting • The Software MacKiev Company.....	60
REALbasic Plug-ins • Einhugur Programming Resources.....	70
REALbasic • REAL Software, Inc.....	82
REALbasic Showcase • REAL Software, Inc.....	69-71
Resorcerer • Mathemaesthetics, Inc.....	47
Revolution • Runtime Revolution Limited.....	ifc2
SCM Software • Perforce Software, Inc.....	81
SmallDog.com • Small Dog Electronics.....	17
Software and Hardware • Aladdin Knowledge Systems, Inc.....	35
Software Protection • WIBU-SYSTEMS AG.....	33
Sound Studio • Felt Tip Software.....	23
Stone Studio • Stone Design Corp.....	52
SuSE Linux • SuSE Inc.....	54
SyBase • SyBase, Inc.....	2-3
Timbuktu Pro & netOctopus • Netopia, Inc.....	43
Time Track • Trinifinity Software.....	56
Translation & Localization • Lingo Systems.....	53
Trapcode • Trapcode Software.....	57
UniHelp Module • Electric Butterfly.....	71
Utilities • Sig Software.....	56
Valentina • Paradigma Software.....	27
VXA • Exabyte.....	9
Watson • Karelia Software.....	57
Whistle Blower • James Sentman Software.....	70

The index on this page is provided as a service to our readers. The publisher does not assume any liability for errors or omissions.

To meet deadlines, developers have two choices:

1. Use Perforce
2. Cut Corners



For developers under pressure to manage source code and do more in less time, Perforce's Fast Software Configuration Management System is the must-have tool.

With rival SCM systems, the only way to quicken the pace is to cut corners - but in the long run you pay the price with missed deadlines, uncertain contents, buggy releases and no way back to previous builds.

With Perforce, the fast way is always the right way. Install it fast, learn it fast, execute operations fast. With other SCM systems, developers face an unpleasant choice: do it the right way or do it the fast way. Perforce's speed and reliability mean fast is right. See how Perforce compares with other leading SCM systems at <http://www.perforce.com/perforce/reviews.html>

Run at full speed even with hundreds of users and millions of files. At the core of Perforce lies a relational database with well-keyed tables, so simple operations can be accomplished in near-zero time. Larger operations (like labeling a release and branching) are translated into keyed data access, giving Perforce the scalability that big projects require.

Work anywhere. Perforce is efficient over high-latency networks such as WANs, the Internet and even low-speed dial-up connections. Requiring only TCP/IP, Perforce makes use of a well-tuned streaming message protocol for synchronizing client workspace with server repository contents.

Develop and maintain multiple codelines. Perforce Inter-File Branching™ lets you merge new features and apply fixes between codelines. Smart metadata keeps track of your evolving projects even while they develop in parallel.

Truly cross platform. Perforce runs on more than 50 operating systems, including Windows and nearly every UNIX® variation, from Linux® and Mac OS® X to AS/400 and more.

Integrate with leading IDEs and defect trackers: Visual Studio.NET®, Visual C++®, Visual Basic®, JBuilder®, CodeWarrior®, TeamTrack®, Bugzilla™, ControlCenter®, DevTrack® packages, and more.

PERFORCE
SOFTWARE

Fast Software Configuration Management www.perforce.com

Download your free 2-user, non-expiring, full-featured copy now from www.perforce.com
Free (and friendly) technical support is on hand to answer any and all evaluation questions.

All trademarks used herein are either the trademarks or registered trademarks of their respective owners.

personals

like ships
passing in
the night

SUPERMARTIN Not older gal with Spinal Tap working at WFM. Meets with curly pony tail. I want a hot lunch? #6922

COCOA I did not get you beautiful. I would like to read

PHONE-PLAYING I recently read a book about Georgia #6854

WE MET IN MADISON, summer. Would like to talk again! #6967

FROM WCW. Exchanged letters at cage match. It was pure magic. Would love to get you in a deeper hold. #5827

BEAUTIFUL AND SEVENTEEN. Met you at the Metro. You were on a date with someone else. Next time it will be me. #6973

LEVITATING BUDDHA SWORD-PLAY welding lady. I'm interested but your brave new world expired. Call me at 729

VEGETARIAN BOWLER. You bought me a warm beer and stole my heart. Used same kind of ball and spoke of hatred of rented shoes. Would love to chat over hummus. #5684

LAWN CARE? My husband got lazy and hired you to mow our lawn. Instead you landscaped my erotic fantasies in ways I have never imagined. Could not pronounce your name but looked very sensual. I had blue shoes on. #3696

TWINS WHO SAW TWINS. Us: two handsome guys in suspenders walking Maltese. You: two foxy ladies fighting over last piece of gum. What do you say the four of us make two good looking couples? Twin love. Call me. #4747

DUGOUT FIRECRACKER. You were cleaning up a beer that you spilled on your white t-shirt and threw a whiskey bottle at the umpire. Must meet you and make children. #5551

LORIN, YOU'RE GORGEOUS, funny and brilliant. I don't deserve you but a girl can dream. #6885

SY FROM DOWN SOUTH. You sat with us at Smitty's 11/24, missed you at The Boot. Wanna meet after work sometime? Call and gimme your number. Jenny #6927

CLASSY LATINA With substance, 5'6 plus, non-smoker, very pretty, good fluff, passionate, vibrant, sexual, call me. #6900

RON FROM SANTA FE. You danced with me at the Rattle & Cattle Club. Thanks! I was shy. Can I see you again? Will come down to look for you Friday night. #6841

GORGEOUS, WITTY, BORN TO TEASE: love theater, dance, golf, warm conversation. If you're tall, 35-55, non-smoker, financially secure, enjoys pampering a woman traveling, long walks and stars. please call #6821

ME: LONELY SWEDISH LINGERIE MODEL and gourmet cook. You: slightly overweight and without ambition. Must be into computers, role-playing games and air hockey. #5988

49, PLAIN BUT WITH GOOD BITS. overweight but curvy, great round, wicked sense of humor, and a weird view of life looking for like minded person. Age not important. #6994

TREE HUGGER, MID 50'S, light smoker, tall. Like easy living, tropics and I'm friendly. Seeking considerate, semi-fit companion with a clue. Must love dogs and reggae. #6963

ARE YOU HONEST, handsome, successful, financially secure, intelligent, world traveled, cultured, creative, fit, playful, adventurous, passionate, humorous, caring, loving, and between 46 and 58? Respond to European, blonde female counterpart. #6802

ARE YOU STIMULATED BY beauty, intelligence, humor? Attractive SWF wants good looking SWM or SHM for romantic adventures, possible long term. Essentials: honesty, passion, kindness, sensuality, integrity, open mind. #6741

ATTRACTIVE TALL (5'10"), slender DPWF, 46, emotionally and physically fit, youthful appearance and outlook, intelligent, loving, desires long term relationship. #6853

YOU WON'T BELIEVE YOUR EYES when you see this very cute, petite, DWF, 46, long brown hair/hazel eyes, 5' 110, outgoing personality seeks DWN, 46-55, non smoker, fit, college educated. Call me, let's see if the chemistry is right! #6951

SWF, 28, STRAWBERRY CURLS and tall. Caring, a computer, enlightened, educated, vegetarian. I'm a good cook. Will not see you if you are not a good cook. #6900

NOT SO DESPERATELY seeking one smart, strange, sexy boy to court and spark. Me 23, open to possibilities and ravenous for new life experiences. #6933

SWINGING SANTA. Lonely man who only works 6 weeks a year seeking woman with full time employment with benefits looking to grow old with man who shakes like a bowl full of jelly. #1258

WM, 95, RECENTLY WIDOWED, seeking 18-20 hottie for "fun". Call soon. I'm not getting any younger. I'll put you in my will. #6757

BALD ROMEO. You serenaded the old people at the old people home last weekend. You were a terrible singer and quite unattractive, but your heart is obviously pure gold. My sister would be perfect for you. #7887

MONKEY TRAINER. Seeking woman to train my monkey. Seriously, his name is Murphy and he is a 3 year old chimpanzee. He likes pop lars and nice people. Plus, you and I will have sex. #7874

SINGLE MAN. Single man seeking single woman for relationship. I enjoy dating and talking on the phone to women that I am dating. Would love a chance to date someone. #1254

CUTIE PIE SMARTY PANTS are you ready? I'm 46, 5'6, 120 lbs, blonde, intelligent, educated, vegetarian. I'm a good cook. Will not see you if you are not a good cook. #6900

SIDESH sidesh... well... for... ar...

T ha... ing... light... do you... good... Call me...

MANY WON my life - but no... wonderful woman, smart, professional, at... (non smoker). Love of nature, irreverent humor. #6772

RECENTLY PAROLED, looking for a lady who will keep me on the straight and narrow. Must be into drugs and shoplifting. #6357

HOPLESS ROMANTIC, seeking a lady who will keep me on the straight and narrow. Must be into drugs and shoplifting. #6357

We're Easier.

Create anything from prototypes to full professional applications. Just drag and drop interface elements while REALbasic handles the details. You concentrate on what makes your stuff great — your ideas! REALbasic creates native compiled applications for Macintosh, Mac OS X and Windows without platform-specific adjustments. It's the powerful, easy-to-use tool for creating your own software. Each version of your software looks and works just as it should in each environment.

Complex problems shouldn't require complex solutions. The answer is REALbasic.

 **REALbasic4.5** NEW VERSION

Come see us at Macworld in MacTech Central! Download a free demo. www.realbasic.com